

CENTRAL CIRCULATION BOOKSTACKS

The person charging this material is responsible for its renewal or its return to the library from which it was borrowed on or before the **Latest Date** stamped below. **You may be charged a minimum fee of \$75.00 for each lost book.**

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

TO RENEW CALL TELEPHONE CENTER, 333-8400

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

APR 05 1996

When renewing by phone, write new due date below
previous due date.

L162

Il6r
no. 963
cop. 2
UIUCDCS-R-79-963

UILU-ENG 79 1710
COO-2383-0058

WORKING PAPERS FOR THE 1979 SIGNUM MEETING ON

Numerical Ordinary Differential Equations

R. D. Skeel, Editor

April 1979



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

THE LIBRARY OF THE
MAY 3 1979
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN



Digitized by the Internet Archive
in 2013

<http://archive.org/details/1979signummeetin963sign>

1979 SIGNUM MEETING ON

Numerical Ordinary Differential Equations

April 3-5, 1979 -- University Inn, Champaign, Illinois

jointly sponsored by

Association for Computing Machinery
Special Interest Group on Numerical Mathematics

Department of Computer Science[†]
University of Illinois at Urbana-Champaign

National Science Foundation*

Co-Chairpersons:

C. W. Gear and D. S. Watanabe

Program Committee Chairperson:

R. D. Skeel

R. D. Skeel, Editor

* Meeting supported in part by the National Science Foundation under grant NSF MCS-7820007.

† Printing of these papers was supported in part by the Department of Energy under grant ENERGY/EY-76-S-02-2383.

TABLE OF CONTENTS

SYMPOSIUM

G. Dahlquist, Royal Institute of Technology, Stockholm	
Some properties of linear multistep and one-leg methods for ordinary differential equations	1-1
M. van Veldhuizen, Vrije Universiteit, Amsterdam	
B-stability	2-1
K. Jeltsch, Ruhr-University Bochum, and O. Nevanlinna, Oulu University	
Stability of explicit time discretizations for solving initial value problems	3-1
J. C. Lutchter, University of Auckland	
Stability properties for a general class of methods for ordinary differential equations	4-1
G. Wanner, Université de Genève, S. P. Nørsett, University of Trondheim, and E. Hairer, University of Innsbruck	
A-stability of one- and multistep methods	5-1
S. P. Nørsett, University of Trondheim	
Order and stability of collocation methods with real eigenvalues	6-1
J. D. Lambert, University of Dundee	
Safe point methods for separably stiff systems	7-1
L. D. Thomas, Lawrence Berkeley Laboratory	
The need for improved numerical algorithms in molecular scattering	8-1
M. C. Warming and R. M. Beam, Ames Research Center	
Factored, A-stable, linear multistep methods--an alternative to the method of lines for multidimensions	9-1
R. J. Stetter, Technical University of Vienna	
Tolerance proportionality in ODE-codes	10-1
P. G. Thomsen, Technical University of Denmark	
Numerical solution of large systems of ODE's with sparse Jacobians	11-1
W. A. Enright and M. S. Kamel, University of Toronto	
Automatic partitioning of stiff systems and exploiting the resulting structure	12-1
L. F. Shampine, Sandia Laboratories, Albuquerque	
Implementation of implicit formulas for the solution of ODE's	13-1

G. Corliss, Marquette university, and Y. F. Chang, University of Nebraska--Lincoln Taylor series methods with variable stepsize ratio and variable order	14-1
P. Deuflhard and G. Eader, University of Heidelberg A semi-implicit mid-point rule for stiff systems of ordinary differential equations	15-1
I. E. Hull, University of Toronto What might ODE solvers be tested for?	16-1
I. Gladwell, University of Manchester The NAG library Runge-Kutta routines	17-1
F. E. Cellier and P. J. Moebius, Swiss Federal Institute of Technology, Zurich Towards robust general purpose simulation software	18-1
T. Bar-David, Lockheed, Sunnyvale The A-stability of exponentially fitted linear multi-step methods exact on periodic functions	19-1
S. McKee, Oxford University The linear algebra of discretisation methods	20-1
F. Chipman, Acadia University Some experiments with STRIDE	21-1
A. Iserles, University of Cambridge Exponential interpolations and the numerical solution of O.D.E. (summary)	22-1
R. Dancnick and M. L. Juncosa, Rand Corporation On the existence of maximum polynomial degree Mordsieck-Gear (k,p) methods for all (k,p)	23-1
R. Alfeld, University of Utah Two devices for improving the efficiency of stiff ODE solvers	24-1
S. Fujita, Tokyo Institute of Technology Structural stability and the numerical methods for stiff differential systems	25-1
P. Albrecht, PUC, Rio de Janeiro On the theory and the construction of cyclic multistep methods	26-1
Z. Zlatev, Royal Veterinary and Agricultural University, Copenhagen and P. G. Thomsen, Technical University of Denmark An automatic differential equations solver based on linear multistep methods	27-1

R. L. Brown, University of Virginia Software development for stability analysis of nonlinear differential systems	28-1
J. R. Cash, Imperial College, London Second derivative extended backward differentiation formulae for the numerical integration of stiff systems	29-1
D. D. Greenspan and J. S. Collier, University of Wisconsin--Madison Computer studies of planetary-type evolution	30-1
J. H. Verner, Queen's University On the reliability of error estimators for explicit Runge-Kutta procedures	32-1
J. P. Hennart and R. England, Universidad Nacional Autónoma de México A comparison between several piecewise continuous one step integration techniques	33-1
H. Masr, Military Technical College, Cairo On the A-stability of numerical multistep methods for solving initial value problems in ordinary differential equations with an example for an A-stable multistep method of order 4	34-1
G. J. Cooper, University of Sussex Local error estimates for linear methods for ordinary differential equations	35-1
M. B. Carver, Chalk River Nuclear Laboratories In search of a robust integration algorithm for general library use: some tests, results and recommendations	36-1
F. E. Zadunaisky, CNIÉ-Observatorio Nacional de Física Cosmica and G. Lafferriere, University of Buenos Aires On an iterative improvement of the approximate solution of some ordinary differential equations	37-1
I. F. Chang and R. Morris, University of Nebraska--Lincoln and G. Corliss, Marquette University A portable automatic Taylor series (ATS) compiler for solving ordinary differential equations	38-1
P. Kaps, University of Innsbruck Rosenbrock-type methods for the numerical solution of stiff systems	39-1
D. S. Watkins, Utah State University Determining realistic initial values for stiff systems of ordinary differential equations occurring in ionospheric physics	40-1
J. W. Paine, Australian National University A comparison of methods for computing Sturm-Liouville eigenvalues	41-1



WORKSHOP

Techniques for difficult problems (e.g. oscillatory systems)	no paper
D. G. Bettis, moderator, University of Texas at Austin	
P. Deuflhard, University of Heidelberg	
C. W. Gear, University of Illinois at Urbana-Champaign	
L. R. Petzold, Sandia Laboratories, Livermore	
The numerical solution of second order systems of ODEs	43-1
W. H. Enright, moderator, University of Toronto	
D. G. Bettis, University of Texas at Austin	
P. Deuflhard, University of Heidelberg	
F. T. Krogh, Jet Propulsion Laboratory	
Method of Lines.....	44-1
G. D. Byrne, moderator, University of Pittsburgh	
M. E. Carver, Chalk River Nuclear Laboratories	
A. C. Hindmarsh, Lawrence Livermore Laboratory	
G. K. Leaf, Argonne National Laboratory	
M. Minkoff, Argonne National Laboratory	
W. E. Schiesser, Lehigh University	
R. F. Sincovec, Boeing Computer Services	
Implementation of implicit Runge-Kutta codes	no paper
F. H. Chipman, moderator, Acadia University	
T. A. Eickart, Syracuse University	
J. C. Butcher, University of Auckland	
M. A. Epton, Boeing Computer Services	
S. P. Nørset, University of Trondheim	
L. F. Shampine, Sandia Laboratories, Albuquerque	
Panel on testing	45-1
T. E. Hull, moderator, University of Toronto	
M. E. Carver, Chalk River Nuclear Laboratories	
W. H. Enright, University of Toronto	
I. Gladwell, University of Manchester	
F. T. Krogh, Jet Propulsion Laboratory	
L. F. Shampine, Sandia Laboratories, Albuquerque	
A user interface standard for ODE solvers	46-1
A. C. Hindmarsh, Lawrence Livermore Laboratory	
An incomplete list of better FORTRAN codes for IVPs in ODEs	47-1
R. D. Skeel, University of Illinois at Urbana-Champaign	

CONFERENCE OVERVIEW

R. D. Skeel
Program Committee Chairperson

This meeting is another in a series of single-topic SIGNUM Meetings. It differs from most of its predecessors in two ways. First, it is devoted to a well established and well defined area of numerical mathematics; and although the scope of the meeting is quite narrow, it has a large vertical extent ranging from the very theoretical down to the very practical. Second, by publicizing the meeting well in advance and by obtaining the generous support of the National Science Foundation, it was possible to make the meeting truly international.

The numerical solution of the initial value problem for ordinary differential equations is often cited as one of the notable successes of numerical mathematics. Nevertheless, there are evidently many who believe that much can still be done, and the significant advances being reported at this meeting confirm that belief.

The planning of the program has been influenced to some extent by the following categorization of research efforts in numerical ODEs: (i) theory--the analysis of numerical methods and techniques, (ii) algorithms--the synthesis of numerical methods and techniques, (iii) software--implementation of methods and techniques as general purpose library routines, (iv) applications--design of algorithms and tailoring of software for specific areas of application. These various aspects of numerical ODEs might be visualized as a triangle with algorithms in the center, theory at the top corner, and software and applications at the two supporting bottom corners. A rough breakdown of symposium presentations is fifteen in theory, fourteen in algorithms, five in software, and five in applications. This preponderance of more theoretical papers is balanced by the more practical orientation of the workshop.

The organization of the technical program is in a large part due to the efforts of T. E. Hull, J. D. Lambert, L. F. Shampine, and H. J. Stetter in reviewing the summaries for about thirty-four papers and I would like to thank them very much. Also appreciated is the sponsorship by Tom Hull of the OlympiODE.

I would like to acknowledge the efforts of Bill Gear and Dan Watanabe in the overall organization of the conference as well as their advice on the planning of the program. It should also be noted that Bill organized the workshop. The assistance of Dennis Gannon and Mitch Roth with local arrangements is gratefully acknowledged.

The University of Illinois Department of Computer Science has supported this conference in numerous ways. Barb Armstrong has done a great deal of secretarial work, and business manager Clyde Helm has been very helpful.

Finally, we are very pleased to have so many of the top researchers from throughout the world participate in this meeting. The efforts and expenditures of all participants is much appreciated.

SOME PROPERTIES OF LINEAR MULTISTEP AND ONE-LEG METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS

Germund Dahlquist

Royal Institute of Technology
S-10044 Stockholm, Sweden

1. What is a one-leg method and what relation does it have to a linear multistep method?

This paper deals with the numerical integration of the initial value problem for the differential system,

$$(1.1) \quad dy/dt = f(y), \quad y \in \mathbb{C}^S.$$

To simplify the writing we restrict the discussion to autonomous systems. The generalization of results for non-linear systems to the non-autonomous case is straight-forward, see e.g. Dahlquist (1978).

In the constant stepsize case, $t_n = t_0 + nh$, a one-leg method is written,

$$(1.2) \quad \sum_{j=0}^k \alpha_j y_{n+j} = hf \left(\sum_{j=0}^k \beta_j y_{n+j} \right),$$

where $\sum \beta_j = 1$. It is "The one-leg twin" of the more familiar linear multistep method,

$$(1.3) \quad \sum_{j=0}^k \alpha_j \hat{y}_{n+j} = h \sum_{j=0}^k \beta_j f(\hat{y}_{n+j}).$$

Some one-leg methods have been in use for a long time. One of the simplest examples is the implicit midpoint method,

$$y_{n+1} - y_n = hf \left(\frac{1}{2} (y_{n+1} + y_n) \right),$$

which is the one-leg twin of the trapezoidal method,

$$y_{n+1} - y_n = \frac{1}{2} h (f(y_{n+1}) + f(y_n)).$$

The backward differentiation methods are their own one-leg twins.

In terms of the displacement operator E , $Ey_n = y_{n+1}$, and the generating polynomials,

$$\rho(\zeta) = \sum_{j=0}^k \alpha_j \zeta^j, \quad \sigma(\zeta) = \sum_{j=0}^k \beta_j \zeta^j,$$

the methods can be written,

$$(1.2') \quad \rho(E)y_n = hf(\sigma(E)y_n) \quad (\text{one-leg})$$

$$(1.3') \quad \rho(E)\hat{y}_n = h\sigma(E)f(\hat{y}_n) \quad (\text{linear multistep}).$$

We assume that ρ, σ are relatively prime.

The normalization convention and the consistency conditions read,

$$(1.4) \quad \sigma(1) = 1, \quad (\text{normalization})$$

$$(1.5) \quad \rho(1) = 0, \quad \rho'(1) = \sigma(1) \quad (\text{consistency})$$

For constant stepsize the following relations between the twins are well known, Dahlquist (1975a).

a) *LINEAR IDENTITY*. They yield the same difference equation for a linear autonomous system.

b) *NON-LINEAR EQUIVALENCE* (first part). If $\{y_n\}$ satisfies the one-leg equation (1.2') and if

$$\hat{y}_n = \sigma(E)y_n,$$

then $\{\hat{y}_n\}$ satisfies the linear multistep equation (1.3').

In order to prove the second statement, one multiplies (1.2') by the operator $\sigma(E)$ and makes use of commutativity, $\sigma(E)\rho(E) = \rho(E)\sigma(E)$.

Dahlquist (1975a) introduced *one-leg methods*, since they gave *neater formulations in the analysis of numerical stability for non-linear systems*. There are, however, other reasons for the consideration of these methods. For example, they give an *efficient implementation of the linear multistep twins*. Note that one needs k vectors only,

$y_n, y_{n+1}, \dots, y_{n+k-1}$, in order to compute y_{n+k} from (1.2). Then $\hat{y}_n = \sigma(E)y_n$ is computed but not stored. The most straightforward implementation of (1.3) needs $2k$ values from the past (except for "sparse" methods like Adams or BDF), if one does not recompute k old values of $f(y)$. However, Skeel (1978) has recently found other implementations of a general linear multistep method with the same advantage as the one-leg implementation.

A one-leg method may also be considered as a *numerical method in its own right*. Nevanlinna and Liniger (1979) point out that it is advantageous compared to a linear multistep method in *variable step* computations. On the other hand, the latter has, in a sense, usually higher *accuracy* than its one-leg twin, see Section 2.

In the variable stepsize case, let $h_m = t_m - t_{m-1}$. We temporarily write a one-leg method in the form,

$$\sum_{j=0}^k \alpha_j y_{n+j} = f \left(\sum_{j=0}^k \beta_j y_{n+j} \right),$$

where the α_j, β_j depend on the stepsizes $h_{n+1}, h_{n+2}, \dots, h_{n+k}$. We find, by dimensional analysis, that it is sufficient to consider methods of the form,

$$(1.6) \quad \rho_n(E)y_n = \bar{h}_n f(\sigma_n(E)y_n), \quad \bar{h}_n > 0,$$

where the coefficients of ρ_n, σ_n depend on the ratios h_{n+j}/h_{n+j-1} , $j=2,3,\dots,k$, only, and \bar{h}_n is homogeneous of the first degree in

$h_{n+1}, h_{n+2}, \dots, h_{n+k}$. The consistency conditions read

$$(1.7) \quad \rho_n(1) = 0, \quad \rho_n(E)t_n = \bar{h}_n.$$

The method (1.6) is *normalized* if,

$$(1.8) \quad \rho'_n(1) = \sigma_n(1) = 1.$$

The condition $\rho'_n(1) = \sigma_n(1)$ is introduced for formal convenience only. It means no restriction, since it will just give a natural scaling of \bar{h}_n , by (1.7).

Note that in general, $\rho_n(E)$ and $\sigma_n(E)$ don't commute. Therefore, the short proof of non-linear equivalence given above, does not work for arbitrary changes of stepsize. In fact, it follows from an argument of Nevanlinna and Liniger (1979, p.64) that there is no non-linear equivalence for arbitrary stepsize sequences. However, for *geometric* stepsize sequences, i.e. if

$$(1.9) \quad h_n = h_0 \cdot \tau^n,$$

the polynomials ρ_n, σ_n do not depend on n (although they depend on τ). In this case $\rho_n(E)$ and $\sigma_n(E)$ commute. The following results are proved very much like the special case in Dahlquist (1975a).

PROPOSITION 1. *Linear identity for arbitrary stepsize sequences. For a linear autonomous system the one-leg method (1.6) gives the same difference equation as the linear multistep method,*

$$(1.10) \quad \rho_n(E)y_n = \bar{h}_n \sigma_n(E)f(y_n).$$

Non-linear equivalence for geometric stepsize sequences defined by (1.9). Assume that $\{y_n\}$ satisfies the one-leg difference equation,

$$(1.11) \quad \rho(E)y_n = v_n,$$

and let $\hat{y}_n = \sigma(E)y_n$. Then

$$(1.12) \quad \rho(E)\hat{y}_n = \sigma(E)v_n.$$

If $v_n = \bar{h}_n f(\hat{y}_n)$ we can write (1.12) in either of the forms,

$$(1.13) \quad \rho(E)\hat{y}_n = \sigma(E)(\bar{h}_n f(\hat{y}_n)),$$

$$(1.13') \quad \rho(E)\hat{y}_n = \bar{h}_n \sigma(E)f(\hat{y}_n).$$

Conversely, assume that for $n \geq 0$, \hat{y}_n satisfies (1.12). Let P_m, Q_m be polynomials of degree less than k , such that for some m , $0 \leq m \leq 2k-1$,

$$(1.14) \quad P_m(\zeta)\sigma(\zeta) - Q_m(\zeta)\rho(\zeta) = \zeta^m,$$

and let

$$(1.15) \quad y_n = P_m(E)\hat{y}_{n-m} - Q_m(E)v_{n-m}, \quad n \geq m.$$

Then $\{y_n\}$ is independent of m , and

$$(1.16) \quad \sigma(E)y_n = \hat{y}_n, \quad \rho(E)y_n = v_n.$$

In particular, for $v_n = \bar{h}_n f(\hat{y}_n)$ it follows that if $\{\hat{y}_n\}$ satisfies the linear multistep equation (1.13) then $\{y_n\}$ satisfies the one-leg equation (1.6), and vice versa. ■

COMMENTS.

1) Note that (1.10) and (1.13) are different.

- 2) The sequence v_n is introduced to facilitate the applications to perturbed difference equations needed in the error analysis, see e.g. Dahlquist (1975b).
- 3) Eqn. (1.15) gives for $m=n$, $n=0,1,2,\dots,k-1$, initial values for the above mentioned one-leg implementation of a linear multistep method, with given initial values, \hat{y}_j , $j=0,1,\dots,k-1$.

2. The local error

Let $y(t)$ be an exact solution of the differential system (1.1), and define the local error of the one-leg method (1.2') by,

$$(2.1) \quad \varepsilon_n = \bar{h}_n^{-1} \rho_n(E)y(t_n) - f(\sigma_n(E)y(t_n)).$$

The displacement operator E operates on a sequence space, $Ey_m = (Ey)_m = y_{m+1}$, (not on a function space). Here

$$\rho_n(E)y_m = \sum_{j=0}^k \alpha_{jn} y_{m+j},$$

and similarly for $\sigma_n(E)$. Introduce the operators L_n, S_n ,

$$(2.2) \quad \begin{aligned} L_n \varphi(t_n) &= \rho_n(E)\varphi(t_n) - \bar{h}_n \sigma_n(E)\varphi'(t_n), \\ S_n \varphi(t_n) &= \sigma_n(E)\varphi(t_n) - \varphi(\sigma_n(E)t_n). \end{aligned}$$

S_n is a kind of interpolation error operator. Then

$$(2.3) \quad \begin{aligned} \varepsilon_n &= \bar{h}_n^{-1} L_n y(t_n) + S_n y'(t_n) + f(y(\sigma_n(E)t_n)) - \\ &\quad - f(\sigma_n(E)y(t_n)) = \bar{h}_n^{-1} L_n y(t_n) + S_n y'(t_n) - \\ &\quad - f'_y S_n(y(t_n)) + O(\|S_n y(t_n)\|^2). \end{aligned}$$

For a linear multistep method, the first term only is present. We observe that we must put constraint on the σ -polynomial in order to obtain the same order of accuracy for the one-leg method as for the linear multistep method. Choose a suitable reference point t_n^* for Taylor expansions and put,

$$(2.4) \quad \begin{aligned} t_{n+j} &= t_n^* + \tau_{jn} \bar{h}_n \\ M_{qn}(\rho) &= \sum_{j=0}^k \alpha_{jn} \tau_{jn}^q, \quad M_{qn}(\sigma) = \sum_{j=0}^k \beta_{jn} \tau_{jn}^q, \quad q \geq 0 \end{aligned}$$

(M stands for "moment"). Then,

$$\begin{aligned} \rho_n(E)y_n &= \sum_{j=0}^k \alpha_{jn} y(t_{n+j}) = \\ &= \sum_{j=0}^k \alpha_{jn} \sum_{q=0}^{\infty} (\tau_{jn} \bar{h}_n)^q y^{(q)}(t_n^*)/q! = \\ &= \sum_{q=0}^{\infty} \bar{h}_n^q M_{qn}(\rho) y^{(q)}(t_n^*)/q! \end{aligned}$$

This calculation can easily be modified to cover also the case of non-analytic $y(t)$. Similarly, a straight-forward calculation yields, if we assign an arbitrary finite value to $M_{-1n}(\sigma)$,

$$(2.5) \quad L_n y(t_n) = \sum_{q=0}^{\infty} [M_{qn}(\rho) - q M_{q-1n}(\sigma)] \bar{h}_n^q y^{(q)}(t_n^*)/q!$$

and, if we note that $\sigma_n(E)t_n = t_n^* + \bar{h}_n M_{1n}(\sigma)$,

$$(2.6) \quad S_n \varphi(t_n) = \sum_{q=0}^{\infty} [M_{qn}(\sigma) - M_{1n}^q(\sigma)] \bar{h}_n^q \varphi^{(q)}(t_n^*)/q!.$$

The following result is then easily obtained.

PROPOSITION 2. The order of accuracy of the *linear multistep method* (ρ_n, σ_n) is the largest integer p , such that

$$(2.7) \quad M_{qn}(\sigma) = q \cdot M_{q-1,n}(\sigma), \quad q=0,1,2,\dots,p.$$

The order of accuracy of the corresponding *one-leg method* (ρ_n, σ_n) is $\min(p,r)$, where r is the largest integer for which the following equations hold,

$$(2.8) \quad M_{q-1,n}(\sigma) = M_{1n}^{q-1}(\sigma), \quad q=1,2,\dots,r.$$

The asymptotic form (when $h \rightarrow 0$) for ϵ_n is,

$$(2.9) \quad \epsilon_n \sim \left[M_{p+1,n}(\sigma) - (p+1)M_{pn}(\sigma) \right] \bar{h}_n^p y^{(p+1)} / (p+1)! + \\ + \left[M_{rn}(\sigma) - M_{1n}^r(\sigma) \right] \bar{h}_n^r y^{(r+1)} - f_y' y^{(r)} / r! \quad \blacksquare$$

The derivatives are evaluated at t_n^* , and f_y' is the Jacobian, evaluated at $y(t_n^*)$. Note that the second term in (2.9) vanishes when the system is linear and autonomous, as it should, due to the "linear identity property", see Prop.1.

Also note that for $q=1$, (2.8) becomes the normalization convention $\sigma_n(1)=1$, and that (2.8) is trivial for $q=2$. There are *special constraints* for one-leg methods for $r \geq 3$ only.

The relations (2.8) were first obtained by Liniger (1978), for constant stepsize, after that Warming and Dahlquist had realized that in general the order will not exceed 2 for one-leg methods.

In view of the non-linear equivalence (for constant stepsize) it may seem paradoxical that these methods do not have the same "order of accuracy". However, Prop.1 can only be used to prove that $\sigma(E)y_n - y(\sigma(E)t_n) = O(h^p)$. It does not follow that $y_n - y(t_n) = O(h^p)$. Similar observations have been made for other numerical methods, see Butcher (1973).

3. Stability and contractivity

Consider the application of the method (ρ, σ) with constant step h to the test equation, $y' = \lambda y$, and set $Y_n = (y_n, y_{n+1}, \dots, y_{n+k-1})^T$. The difference equation can then be written in vector-matrix form,

$$(3.1) \quad Y_{n+1} = A(\lambda h) Y_n,$$

where $A(\lambda h)$ is a $k \times k$ companion matrix.

We recall that the *stability set* is

$$S = \left\{ \lambda h \in \mathbb{C} : \{A^n(\lambda h)\}_0^\infty \text{ is bounded} \right\}.$$

The *contractivity set* K , Nevanlinna and Liniger (1978), with respect to a given norm $\|\cdot\|$ in \mathbb{C}^k is $\{\lambda h \in \mathbb{C} : \|A(\lambda h)\| \leq 1\}$. *) Obviously $K \subset S$. We use terms like A-contractivity etc. in obvious analogy with A-stability etc.

We define the *G-norm* of a vector,

$$\|v\|_G^2 = v^H G v, \quad v \in \mathbb{C}^k.$$

Let K_G be the corresponding contractivity set. By the theorem of Stein, there exists, for every fixed $\lambda h \in S$ a matrix G , such that $\|A(\lambda h)\|_G \leq 1$. For most

methods there is no G that can be used in the whole of S , but there exists a useful substitute.

We call a set in \mathbb{C} *circularly bounded* if it is the closure of either a disk or the complement of a disk or a half-plane. It can also be voider a point.

PROPOSITION 3. (Dahlquist 1979, 1978). K_G is circularly bounded. It is symmetric with respect to the real axis, if G is real and symmetric. To any circularly bounded $C \subset S$ there exists a G -norm, such that $K_G \supset C$. \blacksquare

A corollary is that A-stability implies A-contractivity for some G -norm (a property previously called G -stability). Unfortunately, it also follows that for a method that is not A-stable, one cannot find a G such that K_G contains both 0 and ∞ . For other norms K is usually not circularly bounded.

When a one-leg method is applied to the equation $y' = \lambda(t)y$, the difference is still of the form (3.1), and if $\lambda(t)h \in K$ for all t , then $\|A(\lambda t_n)\| \leq 1, \forall n$. For a linear multistep method, the difference equation becomes more complicated. The matrix A depends on the values of λ at several points. This example indicates both why one-leg methods may be easier to analyse than linear multistep methods in more general situations, and why contractivity is easier to generalize than plain stability.

For G -norms, there is a straight-forward generalization to *nonlinear systems with a certain monotonicity property* related to K_G , Dahlquist (1975b, 1978). The theory is given for constant stepsize only, but the extension to arbitrary stepsize sequences offers no difficulty, provided that there is a G that can be used for all occurring (ρ_n, σ_n) . We shall see in Section 4 that there exist one-leg 2-step methods with this property. Methods, which do not possess this property, may be open for an analysis, based on estimation of the rate of change of G .

For *maximum norms*, Nevanlinna and Liniger (1978, 1979) have obtained very interesting results. In non-linear problems they assume a kind of diagonal dominance for the Jacobian. Their analysis is normally restricted to a smaller class of methods than the G -norm theory, but in some respects the maximum norm theory is stronger, e.g. in connexion with A_0 -contractivity. See also the end of Section 4.

4. A-contractivity of two-step methods for arbitrary stepsize sequences

Let

$$\rho_n(\zeta) = \alpha_{2n}\zeta^2 + \alpha_{1n}\zeta + \alpha_{0n},$$

$$\sigma_n(\zeta) = \beta_{2n}\zeta^2 + \beta_{1n}\zeta + \beta_{0n}.$$

Assume that the method is A-stable for each n , i.e.

$$|\zeta| > 1 \Rightarrow \operatorname{Re} \sigma_n(\zeta) / \rho_n(\zeta) > 0, \quad \forall n.$$

For brevity we drop n in the subscripts of the coefficients. It must be remembered, however, that they depend on h_{n+2}/h_{n+1} .

LEMMA 1. The consistent, normalized A-stable two-step methods can be expressed in terms of three non-negative parameters, a, b, c ,

$$(4.1) \quad \begin{aligned} 2\alpha_2 &= c+1, & 4\beta_2 &= 1+b+(a+c), \\ 2\alpha_1 &= -2c, & 4\beta_1 &= 2(1-b), \\ 2\alpha_0 &= c-1, & 4\beta_0 &= 1+b-(a+c). \end{aligned}$$

* In this paper a matrix norm is subordinate to the given vector norm.

Let $a = -a_1$, $b = 1 - \beta_1$, $a + c = \beta_2 - \beta_0$.

Equation (4.1) is obtained by the transformation $\tau = (\tau_2 + \tau_0)/2$. Then $\sigma(\zeta)/\rho(\zeta)$ can be expressed in the form,

$$\sigma(\zeta)/\rho(\zeta) = z + \frac{az+b}{z+c}, \quad a, b, c \geq 0.$$

For put $z = (\tau+1)/(\tau-1)$ and match coefficients, remembering the normalization (1.5).

We shall now use the notations and results of section 2 to express *second order accuracy*. It is convenient to put $t_n^* = t_{n+1}$ and express stepsize changes in terms of the parameter, α , defined thus,

$$(4.2) \quad \alpha = c(\tau_2 + \tau_0)/2.$$

$\alpha > 0$ means increase of stepsize.

We substitute this and (4.1) into (2.7). It is sufficient to consider $q=1$ and $q=2$. We obtain after some computation:

LEMMA 1. Let $\alpha(1-\alpha)c \neq 0$. The method defined by (4.1) is A-stable and second-order accurate, iff

$$(4.3) \quad \frac{ac}{\alpha} + \frac{b}{1-\alpha} = 1 - c^2, \quad a \geq 0, b \geq 0, c > 0.$$

If $\alpha = 0$ the condition reads $a=0$. If $\alpha=1$ and $c > 0$ the condition reads $b=0$. If $c=0$ then $(1-b)(\tau_2+\tau_0) = 2a$, $\tau_2 - \tau_0 = 2$.

In order to facilitate the comparison with a more familiar parameterization, we mention the following relations between (h_{n+2}, h_{n+1}) and (α, \bar{h}_n) , which follow from (4.2) and (2.7),

$$(4.4) \quad \begin{aligned} h_{n+2} &= \tau_2 \bar{h}_n = (1 - \alpha + \alpha/c) \bar{h}_n, \\ h_{n+1} &= -\tau_0 \bar{h}_n = (1 - \alpha - \alpha/c) \bar{h}_n. \end{aligned}$$

It can be shown, Dahlquist (1979), that the only possible G-matrices for the pair (ρ_n, σ_n) defined by (4.1) are of the form (apart from a positive multiplier),

$$(4.5) \quad G = \begin{bmatrix} 1+g-2c & 1-g \\ 1-g & 1+g+2c \end{bmatrix},$$

where

$$(4.6) \quad g > c^2, \quad (g - c^2 - b - ac)^2 \leq 4abc.$$

A variable stepsize method is characterized by the triplet $\{a(\alpha), b(\alpha), c(\alpha)\}$. We want G to be the same for all α , apart from a scalar multiplier.

It follows that g and c are independent of α , hence by (4.1), ρ_n has to be the same all the time. Assume that $c \neq 0$. If we require $b(\alpha)$ to be continuous at $\alpha=0$, then, by (4.3) $a(\alpha)/\alpha$ is continuous at $\alpha=0$, but since $a(\alpha) \geq 0$ everywhere, we conclude that $a(0) = a'(0) = 0$, and hence

$$a(0) = 0, \quad b(0) = 1 - c^2.$$

Then, by (4.5), $g=1$, and hence the only possibility is

$$(4.7) \quad G = \begin{bmatrix} 2(1-c) & 0 \\ 0 & 2(1+c) \end{bmatrix}, \quad 0 < c < 1.$$

This result was earlier obtained by Nevanlinna (1979).

We consider the methods defined by the equations,

$$(4.8) \quad \begin{aligned} \sigma(\zeta) &= (1-c^2)\alpha^2, \\ \rho(\zeta) &= (1-c^2)(1-\alpha)^2, \\ \sigma(\zeta) &= 0 = \text{const.} \quad 0 < c < 1. \end{aligned}$$

It is easily verified that (4.7) is satisfied, and that (4.6) holds with $g=1$. In fact,

$$(4.9) \quad (1 - c^2 - b(\alpha) - a(\alpha)/c)^2 = 4a(\alpha)b(\alpha)/c,$$

and we summarize our results.

PROPOSITION 4. For any constant c , $0 < c < 1$, equations (4.8) and (4.1) define a second-order accurate one-leg method, which is A-contractive with respect to the G-norm (4.7). The parameters α and \bar{h}_n are given by (4.4).

There is a geometric idea behind the method (4.8). In the plane with cartesian coordinates (ac, b) , eqn. (4.3) defines a one-parameter family of straight lines, the envelope of which is given by (4.8). Then (4.9) shows that this envelope is exactly the parabolic boundary of the domain defined by the second inequality (4.6), for $g=1$. This coincidence, which we cannot yet "explain", indicates that (4.8) gives a complete account of methods that satisfy the restrictions formulated in this section. (Alternative methods are perhaps obtained by dropping the continuity requirement for $b(\alpha)$.) It is remarkable that the BDF method, for which $c=2$ when $\alpha=0$, is excluded.

Nevanlinna (1979) has somewhat earlier than we found the same family (although with a very different parametrization) and shown that it is A-contractive with respect to the maximum norm, for arbitrary stepsize sequences. His result is more flexible than ours, since it allows $\rho_n(\zeta)$ to be changed during the computations, which we cannot do. On the other hand our result covers (at the time of writing) a different class of non-linear problems.

The analysis can easily be extended to the case where the stability region is assumed to include a circle tangent to the imaginary axis at the origin. We have done some calculations concerning the unique stable method with $k=2$, $p=3$ and some explicit methods with $p=k=2$, but we must say like Fermat: the paper isn't large enough.

REFERENCES

- Butcher J.C. (1973), *The order of numerical methods for ordinary differential equations*, Math. Comp. 27, 793-806.
- Dahlquist G. (1975a), *On stability and error analysis for stiff non-linear problems*, Dept. Comp. Sci., Roy. Inst. of Techn., Stockholm, Report TRITA-NA-7508.
- Dahlquist G. (1975b), *Error analysis for a class of methods for stiff non-linear initial value problems*, Numerical Analysis, Dundee, Springer Lecture Notes in Mathematics, 506, 60-74.
- Dahlquist G. (1978), *G-stability is equivalent to A-stability*, BIT 18, 384-401.
- Dahlquist G. (1979), *Some contractivity questions for one-leg- and linear multistep methods*, to appear as Dept. Comp. Sci., Roy. Inst. of Techn., Stockholm, Report TRITA-NA-7905.
- Liniger W. (1978), personal communication.
- Nevanlinna O. and Liniger W. (1978, 79), *Contractive methods for stiff differential equations*, Part I, BIT 18, 457-474; Part II, BIT 19, 53-72.
- Nevanlinna O. (1979), personal communication.
- Skeel R.D. (1978), *Equivalent forms of multistep formulas*, Dept. Comp. Sci., Univ. of Illinois at Urbana-Champaign, Report UIUCDCS-R-78-940.

D-STABILITY

M.van Veldhuizen,
Wiskundig Seminarium Vrije Universiteit,
De Boelelaan 1081, 1007MC AMSTERDAM

In this paper a new stability criterion for discretization methods for stiff ordinary differential equations is proposed. This new stability concept takes into account the variation in time of the eigenvectors of the Jacobian matrix of the system. As such the new criterion complements the existing criteria; it does not replace them. The new criterion is applied to multistep methods, a generalized Runge-Kutta method, and Galerkin methods (with numerical quadrature). We refer to the new stability criterion as D-stability because of the relationship with the decomposition error.

In the definition of D-stability we consider linear homogeneous systems of the type

$$(1) \quad x' = A(t)x,$$

where $A(t)$ is a matrix and $x' = dx/dt$. The matrix $A(t)$ may be obtained as the Jacobian matrix (evaluated along a smooth solution) related to a non-linear system. We restrict ourselves to discretization methods which discretize the system (1) by the recursion

$$(2) \quad y_{i+1} = G(t_i, h)y_i,$$

where h is the stepsize, $t_i = t_0 + ih$, and where $G(t_i, h)$ is a matrix. If the discretization method is a one-step method, then y_i is the approximation to $x(t_i)$. If the discretization is a multistep method, then the recursion (2) is the formal one-step recursion corresponding to the multistep recursion and the meaning of y_i has to be changed accordingly.

For stiff systems it is very difficult to obtain stability for practical values of the stepsize h . In order to obtain valuable information about discretization methods for stiff systems for realistic values of the stepsize G.G. Dahlquist restricted himself to the scalar model equation $x' = \lambda x$ by introducing his concept of A-stability, cf. [2]. This concept has been modified in various ways, but in all modifications the time-dependence of the eigenvectors of the Jacobian matrix is neglected. However, the time-dependence of the eigenvectors of the Jacobian matrix cannot always be neglected in the analysis of stiff discretized problems. The trapezoidal rule provides an example. For this method the source of the difficulties is the bad decomposition error. As such the difficulty was known to Dahlquist and Lindberg [4] and Van Veldhuizen [7]. However, no simple criterion for investigating such phenomena in other methods seems to exist. In this paper we propose the concept of D-stability. We define it in such a way that a D-stable method does not suffer from a bad decomposition error for a large class of stiff problems.

Definition. The discretization method resulting in the recursion (2) is called D-stable if for all t in the interval, all $h \in (0, h_0]$ and all stiff systems of a class \mathcal{J} (see below) $\|G(t, h)\| \leq M$, M a constant independent of h and the system in the class \mathcal{J} .

Consequently, D-stability is a weak concept as only the transition in one step should be bounded uniformly

in the stepsize h . On the other hand, D-stability is a strong concept by requiring this uniformly in a class of stiff systems. Obviously, the concept of D-stability complements, but not replaces, the concept of A-stability and related ones. Now we come to the description of the class \mathcal{S} of stiff systems. This class \mathcal{S} should be sufficiently large because otherwise the concept of D-stability becomes a triviality. On the other hand, we aim at a simple criterion and so we must be restrictive in the choice of the class \mathcal{S} . The class \mathcal{S} as described below seems adequate.

Class \mathcal{S} . The class \mathcal{S} consists of all linear systems of the type $x' = A(t)x$ satisfying the conditions:

$$(S1) \quad x(t) \in \mathbb{R}^2.$$

$$(S2) \quad A(t) = T(t)\Lambda(t)T(t)^{-1} \quad \text{for all } t. \text{ Here } \Lambda(t) \text{ is a diagonal matrix, } \Lambda(t) = \text{diag}\{ \lambda(t)/\epsilon, \mu(t) \}. \text{ Also } \text{Re}(\lambda(t)) \leq \underline{\lambda} < 0, \text{ and } \epsilon \in (0, \epsilon_0].$$

$$(S3) \quad \lambda, \mu, T \text{ and } T^{-1} \text{ depend smoothly on } t \text{ and } \epsilon, \text{ and the derivatives from order zero up to a certain order (sufficiently high) are bounded uniformly in } t \text{ and } \epsilon \in (0, \epsilon_0].$$

Obviously, the systems in the class \mathcal{S} are stiff. In many cases it is realistic to consider a subset of the class \mathcal{S} . E.g. one might consider systems with real eigenvalues only. Also, it is possible to restrict the class \mathcal{S} to the set of stiff systems with a particular type of coupling (strong, weak) between the smooth component and the transient one. In this survey we do not go into these details.

The stiffness is introduced in the class \mathcal{S} by the eigenvalue $\lambda(t)/\epsilon$. We must have $\epsilon \in (0, \epsilon_0]$ in the class \mathcal{S} because otherwise the definition of D-stability becomes a trivial one. We now describe some of the results; we look at three classes of discretization methods.

Multistep Methods. A multistep method defined by the generating polynomials ρ and σ of degree $\leq k$ is D-stable if and only if $\sigma(z) = \beta_0 z^k$. This result once more justifies the choice of the backward differentiation methods in the code by Gear[5]. However, the one-leg method (see Dahlquist[3]) corresponding to the multistep method is D-stable if the method is A_0 -stable. This result emphasizes the importance of one-leg methods as introduced by Dahlquist[3].

Generalized Runge-Kutta Methods. Verwer[8] introduces the concept of internal S-stability. Roughly speaking, a generalized Runge-Kutta method is internally S-stable if it has good damping properties at each stage when applied to the model $x' = \lambda x$. It turns out that the internally S-stable methods of Verwer[8] are not D-stable. This shows that D-stability is not a consequence of good damping properties.

Galerkin Methods. Here we restrict ourselves to Galerkin methods with approximate quadrature as considered (among others) by Axelsson[1] and Weiss[9]. With suitable Gaussian quadrature or Radau quadrature these methods reduce to collocation methods, while the corresponding method with Lobatto quadrature is not a collocation method. We have the following result: all A-stable collocation methods are D-stable; the method with Lobatto points is not D-stable. This result is important in view of the use of such methods in

REFERENCES

- [1] Axelsson, O.: A Class of A-stable Methods. Nordisk Tidskr. Informationsbehandling (BIT) 9, 185-199 (1969)
- [2] Dahlquist, G.G.: A Special Stability Problem for Linear Multistep Methods. Nordisk Tidskr. Informationsbehandling (BIT) 3, 27-43 (1963)
- [3] Dahlquist, G.G.: Error Analysis for a class of Methods for Stiff Non-linear Initial Value Problems. Lecture Notes in Mathematics, vol. 506, pp. 60-72. Berlin-Heidelberg-New York: Springer 1976
- [4] Dahlquist, G.G., Lindberg, B.: On some Implicit One-Step Methods for Stiff Differential Equations. TRITA-NA-7302, Dept of Information Processing, Royal Institute of Technology, Stockholm.
- [5] Gear, C.W.: Algorithm 407, DIFSUB for solution of ordinary differential equations. C.A.C.M., 14, 185-190 (1971)
- [6] Russell, R.D., Christiansen, J.: Adaptive Mesh Selection Strategies for Solving Boundary Value Problems. Siam J. Numer. Anal. (to appear)
- [7] Veldhuizen, M. van: Convergence of One-Step Discretizations for Stiff Differential Equations (thesis). Mathematical Institute, University of Utrecht, Netherlands (1973)
- [8] Verwer, J.G.: S-stability properties for generalized Runge-Kutta Methods. Numer. Math. 27, 359-370 (1977)
- [9] Weiss, R.: The Application of Implicit Runge-Kutta methods to Boundary Value Problems. Math. Comput. 28, 449-464 (1974)

Stability of explicit time discretizations for
solving initial value problems

Rolf Jeltsch, Institute of Mathematics, Ruhr-University Bochum,
D-4630 Bochum, Fed. Rep. of Germany

Olavi Nevanlinna, Department of Mathematics, Oulu University,
SF-90101 Oulu 10, Finland

Stability regions of explicit "linear" time discretization methods for solving initial value problems are treated. A discretization is called "linear" if the right hand side of the differential equation, its total derivatives and iterates of these occur only linearly. Thus the class of methods considered includes Kunge-Kutta-methods, linear multistep methods, multistep methods using higher derivatives, predictor-corrector methods and cyclic methods. If an integration method needs m function evaluation per time step, then we scale the stability region by dividing by m , and show that for any two methods, satisfying some reasonable conditions, the boundaries of the scaled stability regions necessarily intersect. Using this comparison result bounds for the size of the stability regions for three different purposes are given: 1) for "general" nonlinear ordinary differential systems 2) for systems obtained from parabolic problems and 3) for systems obtained from hyperbolic problems.

First we consider the size of disks one can have inside the stability region with origin on their boundary. In the scaled sense the largest disk is obtained with the Euler's method and for any smaller disk there exist explicit linear k -step methods for order $p = k$ which are stable in such a disk.

For parabolic problems one is interested in methods which are stable for long intervals of the form $[-a, 0]$, $a > 0$. It is shown that for explicit linear multistep methods of order $p = k$ the length of the interval is always bounded by 2, but already for $p = k - 1$ one can construct methods which are stable in bounded sectorial sets for arbitrarily big radius and any opening less than π . Hence, although an explicit method cannot be A-stable we can have approximately

A-stable methods with high order.

For hyperbolic problems one is interested in methods with long stability intervals along the imaginary axis. It is shown that in the scaled sense, the leap-frog method has the largest interval and that for any shorter interval $k \in \{2, 3, 4\}$ there are explicit linear k -step methods which are stable on that interval and of order $p = k$. However for $k \equiv 1 \pmod{4}$ an explicit linear k -step method with $p = k$ has no nontrivial interval of stability on the imaginary axis.

By the previous result about the largest stability disk with the origin on its boundary it follows that the radius of such a disk is bounded by m for predictor-corrector methods, where the corrector is iterated $m - 1$ times and the function is evaluated m times. It is shown that this disk can be obtained only if one uses different correctors at each iteration step. When one, as is customary, iterates a fixed corrector, then as m tends to infinity the maximal radius does not tend to infinity, and in fact it is shown that in the limit the maximal radius is 2.

STABILITY PROPERTIES FOR A GENERAL CLASS OF METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS

J.C. Butcher,
University of Auckland,
Auckland, New Zealand

Abstract The concepts of G-stability for linear multistep methods and B-stability for Runge-Kutta methods are combined in a unified approach to non-linear stability for a class of methods general enough to include these as special cases. Recent work on these developments, by Kevin Burrage and the author, is reviewed and, to exemplify the new ideas and techniques, an algebraic stability analysis is presented for the backward differentiation methods of orders 1, 2 and 3.

Introduction The property of A-stability, long accepted as a desirable condition for methods to possess if they are to be used with stiff problems, gives only limited guidance to the numerical behaviour if these problems are also non-linear. For this reason, there has grown up an interest in non-linear problems of the form $y'(x) = f(y(x))$, where y has values in an inner-product space X , which are such that

$$\langle u-v, f(u)-f(v) \rangle \leq 0,$$

for all $u, v \in X$. For such a problem, an immediate consequence is that for two particular solutions y and z , $\|y(x)-z(x)\|$ is a non-increasing function

of x . It is interesting to know whether a particular numerical method exhibits similar behaviour in the results it generates.

This is simply expressed for one step methods as the requirement that for two solution sequences $\dots, y_{n-1}, y_n, \dots$ and $\dots, z_{n-1}, z_n, \dots$, it holds that $\|y_n - z_n\| \leq \|y_{n-1} - z_{n-1}\|$. For multistep methods, in which r vectors in X are used to represent the solution information at the end of a step, it is necessary to construct a norm on X^r using the tensor product of the inner-product on X that comes with the problem and a new inner-product on \mathbb{R}^r that depends on the method. Bounds on the behaviour from step to step can then be expressed in terms of the norm

formed from the tensor product.

If G is the matrix of the inner-product in \mathbb{R}^r , which is characteristic of the method, the corresponding stability property is known as G -stability and was first used by G. Dahlquist [5]. The corresponding property of Runge-Kutta methods was first proposed by the present author [4] and, in collaboration with Kevin Burrage, was further developed [1] and combined with the ideas of G -stability to give the algebraic stability property for general linear methods [2]. This paper is partly a review of the joint work by Dr. Burrage and the author, and partly a study, from the viewpoint of algebraic stability, of three special methods. These methods, the backward differentiation methods of orders 1, 2 and 3, are important components of many algorithms in present use for solving stiff problems.

Partitioned general linear methods.

Let A_{12} be an $s \times r$ matrix, A_{22} an $r \times r$ matrix, B_{11} an $s \times s$ matrix and B_{21} an $r \times s$ matrix. A partitioned general linear method is a general linear method (or (A,B) method) [3], such that A and B are each $(s+r) \times (s+r)$ matrices partitioned as follows

$$A = \begin{bmatrix} 0 & A_{12} \\ 0 & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & 0 \\ B_{21} & 0 \end{bmatrix},$$

where, of course, the various 0 matrices are not necessarily the same sizes. We will, for convenience, denote such a method by writing the non-trivial parts of its coefficient matrices together in the format

$$\begin{array}{c|c} A_{12} & B_{11} \\ \hline A_{22} & B_{21} \end{array}$$

For such a method, when used to solve the equation $y'(x) = f(y(x))$ with step size h , the $s+r$ vectors which represent all the quantities computed in step n , say

$$Y_1^{(n)}, Y_2^{(n)}, \dots, Y_s^{(n)}, y_1^{(n)}, y_2^{(n)}, \dots, y_r^{(n)}$$

are related to the previous step by

$$(2) \quad Y_i^{(n)} = \sum_{j=1}^r a_{ij}^{(12)} y_j^{(n-1)} +$$

$$h \sum_{j=1}^s b_{ij}^{(11)} f(Y_j^{(n)}), \quad i=1,2,\dots,s,$$

$$(3) \quad y_i^{(n)} = \sum_{j=1}^r a_{ij}^{(22)} y_j^{(n-1)} +$$

$$h \sum_{j=1}^s b_{ij}^{(21)} f(Y_j^{(n)}), \quad i=1,2,\dots,r,$$

where $a_{ij}^{(12)}$, $a_{ij}^{(22)}$, $b_{ij}^{(11)}$, $b_{ij}^{(21)}$ are typical elements in A_{12} , A_{22} , B_{11} , B_{21} respectively.

The vectors $Y_1^{(n)}, Y_2^{(n)}, \dots, Y_s^{(n)}$ can be interpreted as s stages, as in a Runge-Kutta method. We see from (2) and (3) that no use in later steps is made of these quantities once the step is completed. On the other hand, $y_1^{(n)}, y_2^{(n)}, \dots, y_r^{(n)}$ represent the r quantities needed at the end of a step to allow the computation of the next step.

Although we have used the symbol $y_i^{(n)}$ for one of these pieces of information, it is not necessarily an approximation to the solution value near the n^{th} integration point x_n . Rather, it should be thought of as an approximation to $u_i y(x_n) + h v_i y'(x_n)$ where u_i, v_i are numbers characteristic of the method.

The special case $r=1$, with A_{22} equal to 1, gives s stage Runge-Kutta methods, whereas

$$\begin{array}{c|c} A_{12} & B_{11} \\ \hline A_{22} & B_{21} \end{array} =$$

$$\begin{array}{cccccccc|c} \alpha_1 & \alpha_2 & \dots & \alpha_k & \beta_1 & \beta_2 & \dots & \beta_k & \beta_0 \\ \hline \alpha_1 & \alpha_2 & \dots & \alpha_k & \beta_1 & \beta_2 & \dots & \beta_k & \beta_0 \\ 1 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ \cdot & 1 & & \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot & \cdot & & \cdot & \cdot \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 0 \\ \cdot & \cdot & & \cdot & \cdot & 1 & & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot & \cdot & & \cdot & \cdot \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \end{array}$$

gives the k-step method

$$y_n = \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \dots + \alpha_k y_{n-k} + h(\beta_0 f(y_n) + \beta_1 f(y_{n-1}) + \dots + \beta_k f(y_{n-k})),$$

$$\begin{aligned} \text{in which } Y_1^{(n)} &= y_1^{(n)} = y_n, \\ y_2^{(n)} &= y_{n-1}, \dots, y_k^{(n)} = y_{n-k+1}, \\ y_{k+1}^{(n)} &= hf(y_n), y_{k+2}^{(n)} = hf(y_{n-1}), \dots, \\ y_{2k}^{(n)} &= hf(y_{n-k+1}). \end{aligned}$$

A non-linear test problem. Let $(X, \langle \cdot, \cdot \rangle)$ denote a real pseudo inner-product space. We recall that an inner-product $\langle \cdot, \cdot \rangle$ on X is a bilinear functional such that $\langle u, v \rangle = \langle v, u \rangle$ for $u, v \in X$ and such that for non-zero $u \in X$, $\langle u, u \rangle > 0$. In a pseudo inner-product this inequality is weakened to $\langle u, u \rangle \geq 0$. Let $\|\cdot\|$ be the corresponding pseudo norm defined by $\|u\| = \langle u, u \rangle^{1/2}$ for all $u \in X$.

Let $f: X \rightarrow X$ be continuous and such that, for all $u \in X$

$$(4) \quad \langle u, f(u) \rangle \leq \varepsilon \|u\|^2,$$

where ε is a real number. A consequence of (4) is that any solution y to the differential equation $y'(x) = f(y(x))$, is bounded in pseudo norm by a multiple of $\exp(\varepsilon x)$. To verify this, we need only differentiate $\|\exp(-\varepsilon x)y(x)\|^2 = \exp(-2\varepsilon x)\langle y(x), y(x) \rangle$ to obtain a zero upper bound on the derivative.

This rather artificial test problem can be used to study the behaviour of the difference of two solutions, as in the G-stability theory, even when the differential equation under study is non-autonomous.

If G is a non-zero non-negative symmetric $r \times r$ matrix, then we can define a pseudo inner-product on X^r using the elements g_{ij} ($i, j=1, 2, \dots, r$) of G , and an existing pseudo inner-product on X . Denoting the pseudo inner-product on X^r by $\langle \cdot, \cdot \rangle_G$, we define its value for $u = (u_1, u_2, \dots, u_r)$, $v = (v_1, v_2, \dots, v_r)$ by

$$\langle u, v \rangle_G = \sum_{i,j=1}^r g_{ij} \langle u_i, v_j \rangle.$$

We will use the pseudo norm $\|\cdot\|_G$, defined from $\langle \cdot, \cdot \rangle_G$, as the measure of the size of a vector in X^r with a view to obtaining bounds on the behaviour of $(y_1^{(n)}, y_2^{(n)}, \dots, y_r^{(n)})$, computed using (1), in terms of the behaviour at the previous step. We will attempt to find bounds that correspond in some way to bounds on the behaviour of the solution to the underlying problem.

Algebraic stability and its generalizations. A partitioned general linear method (1) is said to be (k, l) algebraically stable for k a positive number and l an arbitrary number, if there is a non-zero non-negative symmetric matrix G , and a non-negative diagonal matrix D , such that the (symmetric) matrix

(5)

M =

$$\begin{bmatrix} kG - A_{22}^T G A_{22} & A_{12}^T D - A_{22}^T G B_{21} \\ -2\ell A_{12}^T D A_{12} & -2\ell A_{12}^T D B_{11} \\ D A_{12} - B_{21}^T G A_{22} & D B_{11} + B_{11}^T D - B_{21}^T G B_{21} \\ -2\ell B_{11}^T D A_{12} & -2\ell B_{11}^T D B_{11} \end{bmatrix}$$

is non-negative.

It is a principal result in Burrage and Butcher [2], that if a method satisfies this condition and if a problem satisfies (4) with $\ell = h^*$, then $y^{(n)} = (y_1^{(n)}, y_2^{(n)}, \dots, y_r^{(n)})$ can be (pseudo-) estimated by

$$\|y^{(n)}\|_G \leq k^{1/2} \|y^{(n-1)}\|_G$$

We are interested in obtaining for a method a (k, ℓ) algebraic stability result in which k and ℓ are functionally related by $\varphi(\ell) = k^{1/2}$. To the extent that φ is a good approximation to the exponential function, we obtain behaviour for the computed results that is a good model for the behaviour of the exact result.

In particular, if a method is $(1, 0)$ algebraically stable it will be called, simply, algebraically stable. In addition to this non-linear generalization of A-stability, we are also interested in knowing whether, given k as an arbitrarily small positive number, (k, ℓ) algebraic stability can be obtained for sufficiently small ℓ (generalizing the "behaviour at infinity" aspect of L-stability), and whether we can obtain $(1, \ell)$ stability for sufficiently small ℓ (generalizing part of the definition of stiff stability).

Finally, in this section, we note that, if B_{11} is non-singular (as is typically the case in methods suitable for stiff problems), a congruence transformation can be applied to M given by (5), to yield a symmetric matrix of a simpler form for many methods. If $\tilde{A}_{22} = A_{22} - B_{21} B_{11}^{-1} A_{12}$ and the transforming matrix is

$$P = \begin{bmatrix} I & 0 \\ -B_{11}^{-1} A_{12} & I \end{bmatrix},$$

then $\tilde{M} = P^T M P$ is given by

(6)

 $\tilde{M} =$

$$\begin{bmatrix} kG - \tilde{A}_{22}^T G \tilde{A}_{22} & -A_{12}^T D - \tilde{A}_{22}^T G B_{21} \\ -D A_{12} - B_{21}^T G \tilde{A}_{22} & D B_{11} + B_{11}^T D - B_{21}^T G B_{21} \\ & -2\ell B_{11}^T D B_{11} \end{bmatrix}$$

Where \tilde{M} takes on a simpler form than M , we would use its non-negativity as a criterion for (k, ℓ) algebraic stability.

Application to backward differentiation methods. The backward differentiation methods of orders 1, 2 and 3 respectively, given by

$$(7) \quad y_n = y_{n-1} + hf(y_n),$$

$$(8) \quad y_n = \frac{4}{3}y_{n-1} - \frac{1}{3}y_{n-2} + \frac{2}{3}hf(y_n),$$

$$(9) \quad y_n = \frac{18}{11}y_{n-1} - \frac{9}{11}y_{n-2} + \frac{2}{11}y_{n-3} + \frac{6}{11}hf(y_n),$$

are the first three members of a sequence of important methods for stiff problems. The order 1 and 2 methods are known to be A-stable whereas the 3rd order method given by (9) has a stability region that includes all points in the complex plane with real parts less than $-1/12$.

As partitioned general linear methods they are given respectively by

$$\begin{array}{c|c} 1 & 1 \\ \hline 1 & 1 \end{array}, \begin{array}{c|c} \frac{4}{3} & \frac{-1}{3} \\ \hline \frac{4}{3} & \frac{-1}{3} \end{array} \begin{array}{c|c} \frac{2}{3} \\ \hline \frac{2}{3} \end{array} \text{ and } \begin{array}{c|c} \frac{18}{11} & \frac{-9}{11} \\ \hline \frac{18}{11} & \frac{-9}{11} \end{array} \begin{array}{c|c} \frac{2}{11} \\ \hline \frac{2}{11} \end{array} \begin{array}{c|c} \frac{6}{11} \\ \hline \frac{6}{11} \end{array};$$

$$\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{array}$$

their \tilde{M} matrices (as given by (16)), with D set arbitrarily equal to 1 in each case, are

$$\begin{bmatrix} kg_{11} & -1 \\ -1 & 2 - g_{11} - 2\ell \end{bmatrix},$$

$$\begin{bmatrix} kg_{11} - g_{22} & kg_{12} & \frac{-4}{3} - \frac{2}{3}g_{12} \\ kg_{12} & kg_{22} & \frac{1}{3} \\ \frac{-4}{3} - \frac{2}{3}g_{12} & \frac{1}{3} & \frac{4}{3} - \frac{4}{9}g_{11} - \frac{8}{9}\ell \end{bmatrix}$$

and

$$\begin{bmatrix} kg_{11} - g_{22} & kg_{12} - g_{23} & kg_{13} & \frac{-18}{11} - \frac{6}{11}g_{12} \\ kg_{12} - g_{23} & kg_{22} - g_{33} & kg_{23} & \frac{9}{11} - \frac{6}{11}g_{13} \\ kg_{13} & kg_{23} & kg_{33} & \frac{-2}{11} \\ \frac{-18}{11} - \frac{6}{11}g_{12} & \frac{9}{11} - \frac{6}{11}g_{13} & \frac{-2}{11} & \frac{12}{11} - \frac{36}{121}g_{11} - \frac{72}{121}\ell \end{bmatrix}$$

Not surprisingly, determination of which (k, ℓ) pairs give (k, ℓ) algebraic stability, is trivial for the method given by (7) and progressively more complicated for (8) and (9).

Theorem 1. The first order backward differentiation method is (k, ℓ) algebraically stable whenever $k^{1/2} = 1/(1-\ell)$

Proof. Choose $g_{11} = k^{-1/2}$.

Theorem 2. The second order backward differentiation method is (k, ℓ) algebraically stable for (k, ℓ) related so that

$$(10) \quad \ell = \frac{1}{2} - \frac{1}{2k}, \quad 0 \leq \ell \leq 1,$$

$$(11) \quad \ell = \frac{3}{2} - 2k^{-1/2} + \frac{1}{2}k^{-1}, \quad k \geq 1,$$

or, what is equivalent,

$$k = 1/(1-2\ell), \quad \ell \leq 0,$$

$$k^{1/2} = 1/(2-(1+2\ell)^{1/2}), \quad \ell \geq 0.$$

Proof. We will show that, with ℓ related to k as in (10) and (11), g_{11}, g_{12}, g_{22} may be chosen so that \tilde{M} is of rank 1 with non-negative diagonal elements. Let $\alpha = kg_{12}$, $\beta = kg_{22}$ so that \tilde{M} being of rank 1 would require that $g_{11} = \alpha^2/\beta k + \beta/k^2$, $g_{12} = \alpha/k$, $g_{22} = \beta/k$ (implying that G is positive if $\beta > 0$) and furthermore that

$$(12) \quad \alpha + \beta(4+2\alpha/k) = 0,$$

$$(13) \quad (12-8\ell)\beta = 4\alpha^2/k + 4\beta^2/k + 1.$$

Since the values of ℓ given in (10) and (11) satisfy the inequality $12 - 8\ell > 0$, any real solution to (12) and (13) will necessarily have $\beta > 0$ (because the right hand side of (13) is positive). Hence, it remains to show that, with the given value of ℓ , (12) and (13) do in fact have a real solution pair. If β is eliminated, the resulting equation for α can be written in the form

$$(14) \quad 2(\alpha+k)^4 + (\alpha+k)^2(k-k^2-2\ell k^2) + (k^3-k^4+2\ell k^4) = 0$$

If $k \leq 1$, the value of ℓ given by (10) is satisfied by $\alpha + k = 0$. If $k \geq 1$, the value of ℓ given by (11) is

such as to make (14) a perfect square and the negativity of the coefficient of $(a+k)^2$ guarantees the existence of a positive root for $(a+k)^2$.

Theorem 3. The third order backward differentiation method is (k, t) algebraically stable for (k, t) related so that

$$(15) \quad t = \frac{17}{24} - \frac{3}{4k} - \frac{1}{24} \left(\frac{4}{k} - 3 \right)^{3/2}, \quad 0 < k \leq \frac{64}{49},$$

$$(16) \quad t = \frac{11}{6} + \frac{3}{2k} - k^{-3/2} \left(3k + \frac{1}{3} \right), \quad k \geq \frac{64}{49}.$$

Proof. As in the proof of theorem 2, we will show that G may be chosen so that \tilde{M} is of rank 1 with non-negative diagonals. Let $\alpha = kg_{13}$, $\beta = kg_{23}$, $\gamma = kg_{33}$ so that

$$G = \begin{vmatrix} \frac{\alpha^2}{k\gamma} + \frac{\beta^2}{k^2\gamma} + \frac{\gamma}{k^3} & \frac{\alpha\beta}{k\gamma} + \frac{\beta}{k^2} & \frac{\alpha}{k} \\ \frac{\alpha\beta}{k\gamma} + \frac{\beta}{k^2} & \frac{\beta^2}{k\gamma} + \frac{\gamma}{k^2} & \frac{\beta}{k} \\ \frac{\alpha}{k} & \frac{\beta}{k} & \frac{\gamma}{k} \end{vmatrix}$$

and, writing $m = 33 - 18t$, the conditions on the last row and column of \tilde{M} for it to be rank 1 are

$$(17) \quad 9k^2\gamma + 3k\alpha\beta + 3\beta\gamma - \alpha k^2 = 0,$$

$$(18) \quad 9k\gamma + 2k\beta - 6\alpha\gamma = 0,$$

$$(19) \quad k^3 + 9k^2\alpha^2 + 9k\beta^2 + 9\gamma^2 - mk^3\gamma = 0,$$

and we note that, since the values of t given by (15) and (16) imply that $m > 0$, the positivity of γ , if real solution trios (α, β, γ) to (17), (18) and (19) are shown to exist, will be assured. It is convenient to substitute $\alpha = \frac{3}{4}k + \delta$ and to eliminate β from (17), (18) and (19) to obtain the following equations

$$(20) \quad (144\delta - 108k)\gamma^2 + (144k\delta^2 + 63k^3)\gamma - (16k^3\delta + 12k^4) = 0,$$

$$(21) \quad (1296\delta^2 - 1944k\delta + 729k^2 + 144k)\gamma^2 - 16mk^4\gamma + (144k^3\delta^2 + 216k^4\delta + 81k^5 + 16k^4) = 0.$$

Eliminating γ from (20) and (21) leads to an equation that can be written as

$$(22) \quad (a\delta^4 + b\delta^2 + c)^2 - d\delta^2 = 0,$$

where $a = 20736$, $b = 864k(8 - 3k)$, $c = 192k^4m - 63k^3(16 + 81k)$, $d = 65536k^4(km + 27)^2 - 2359296k^3(1 + 9k)^2$. What we will show is that the value of m resulting from t chosen by (15) and (16) leads to a repeated non-negative root for δ^2 in this equation.

In the case $k \geq 64/49$, the choice of t implies that $d = 0$ and that $c = 9k^{5/2}(4 - 9k^{1/2})^2(8 - 7k^{1/2}) \leq 0$.

It turns out that $k < 64/49$ would not allow a positive root for δ^2 to exist if $d = 0$ so we consider the condition that (22) will have repeated roots if $d \neq 0$. This condition is

$$(23) \quad 27ad^2 + d(4b^3 - 144abc) - c(4b^2 - 16ac)^2 = 0$$

and we will require of the solution we obtain to this that, in addition, $d > 0$. With our values of a, b, c and d , (23) can, after considerable manipulation, be written in the form

$$(24) \quad \left(2m - 243 + \frac{27}{k} \right)^2 \times \left((4m - 81 - \frac{54}{k})^2 - 3^2 \left(\frac{4}{k} - 3 \right)^3 \right) = 0$$

with one of the roots given by

$$(25) \quad m = 81/4 + 27/2k + \frac{3}{4} \left(\frac{4}{k} - 3 \right)^{3/2}$$

which leads to a value of t given by (15).

It remains to prove that, with m given by (25), d is positive or, what is equivalent, that

$$m > \frac{-27}{k} + 6k^{-3/2}(1+9k)$$

for $0 < k < 64/49$.

Multiplying by the positive quantity $4k^{3/2}/3$, we find as an equivalent requirement that

$$(26) \quad 27k^{3/2} + 54k^{1/2} - 8(1+9k) + (4-3k)^{3/2} > 0$$

$$\text{Since } 27k^{3/2} + 54k^{1/2} - 8(1+9k) \\ = (3k^{1/2}-4)(3k^{1/2}-(2+\sqrt{2}))(3k^{1/2}-(2-\sqrt{2})),$$

(26) is clearly satisfied for

$k \in [(2-\sqrt{2})/3, (2+\sqrt{2})/3]$ and, since the left hand side of (26) vanishes when $k=0$ or $k = 64/49$, it will be sufficient to prove that the derivative of this expression is positive when $k^{1/2} \in (0, (2-\sqrt{2})/3)$ and negative when $k^{1/2} \in ((2+\sqrt{2})/3, 8/7)$.

The value of this derivative is

$$(27) \quad \frac{81}{2} k^{\frac{1}{2}} + 27k^{-\frac{1}{2}} - 72 - \frac{9}{2}(4-3k)^{\frac{1}{2}}$$

and, when $k^{1/2} \in (0, (2-\sqrt{2})/3)$, this is bounded below by

$$\frac{81}{2} \frac{2-\sqrt{2}}{3} + 27 \frac{3}{2-\sqrt{2}} - 72 - \frac{9}{2}(4-0)^{\frac{1}{2}} > 0$$

because $\frac{81}{2}k^{1/2} + 27k^{-1/2}$ is easily verified to be an increasing, and

$-\frac{9}{2}(4-3k)^{1/2}$ a decreasing, function in the interval in question. For $k^{1/2} \in ((2+\sqrt{2})/3, 8/7)$, $\frac{81}{2}k^{1/2} + 27k^{-1/2}$ is an increasing function and we bound (27) by substituting $k = 64/49$ to obtain $-27/8 < 0$.

A final remark: the backward differentiation method with order 3 is $(1, -\frac{1}{12})$ algebraically stable and it is noted with interest that the abscissa $-\frac{1}{12}$ exactly agrees with what is obtained in a linear (stiff-stability) analysis.

Acknowledgements. Much of this work was done while the author was visiting the Department of Mathematics at Acadia University and the Department of Mathematics and Statistics at Queen's University. He is indebted to his hosts in these two institutions for their assistance and hospitality and for access to computing facilities.

REFERENCES

- [1] Kevin Burrage and J.C. Butcher, Stability criteria for implicit Runge-Kutta methods, SIAM J. Numer. Anal. (to appear).
- [2] Kevin Burrage and J.C. Butcher, Non-linear stability of a general class of differential equation methods, Computational Mathematics Report No. 18, (1979), University of Auckland.
- [3] J.C. Butcher, On the convergence of numerical solutions to ordinary differential equations, Math. Comp 20 (1966), 1-10.
- [4] J.C. Butcher, A stability property of implicit Runge-Kutta methods, BIT 15 (1975), 358-361.
- [5] G. Dahlquist, On stability and error analysis for stiff non-linear problems, I., Department of Information Processing Report No. NA 75.08, (1975), Royal Institute of Technology, Stockholm.

A-STABILITY OF ONE- AND MULTISTEP METHODS

Gerhard WANNER

Abstract. This work with S.P.Nørsett and E. Hairer deals with a beautiful and geometric discussion of A-acceptability properties of stability functions arising from one- and multi-step methods. We give many necessary or sufficient conditions for A-stability and prove the conjecture of Ehle (1969) as well as the conjecture of Daniel and Moore (1970).

1. Properties of Order Stars.

Consider a rational stability function of a one-step method

$$(1) \quad R(z) = \frac{P(z)}{Q(z)} \quad \begin{array}{l} \deg(P) = a \text{ (nb. of zeros)} \\ \deg(Q) = \omega \text{ (nb. of poles)} \end{array}$$

with the order condition

$$(2) \quad e^z - R(z) = e_{p+1} z^{p+1} + e_{p+2} z^{p+2} + o(z^{p+3}).$$

R is said to be A-acceptable (or the method is A-stable) if $|R(z)| \leq 1$ on the entire left half-plane \mathbb{C}^- .

A typical form of the stability domain

$$(3) \quad D = \{z; |R(z)| \leq 1\}$$

is sketched in Fig.1 (for the 8-1 Padé approximation).

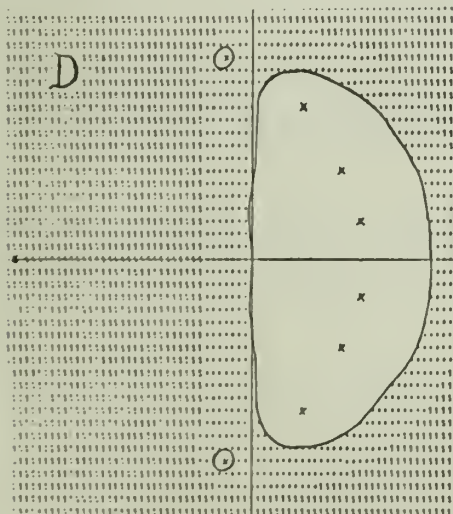


Fig.1. Example of stability domain.

The form of these stability domains is clumsy and not very instructive. So the first treatments on A-acceptability have been loaded with difficult computations. In contrary, the set

$$(4) \quad A = \{z; |R(z)| > |e^z|\}$$

for the same rational approximation is sketched in Fig.2 and gives much more insight (like a hand under an X-ray screen). At every point where R(z) has a contact of order p with $\exp(z)$, the set A consists of p+1 regularly distributed fingers, so we call it the order star of R.

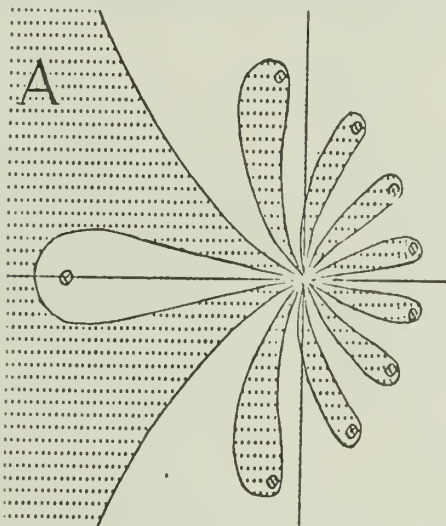


Fig.2. Example of an order star.

The following properties are consequences of simple complex analysis:

Lemma 1. On the imaginary axis, A and D are complementary. So we can have A-acceptability only if A does not meet $i\mathbb{R}$.

Reason: Because of $|e^{iy}| = 1$ for y real.

Lemma 2. For $z \rightarrow 0$, A behaves like a star with p+1 "fingers" of equal width $\pi/(p+1)$, separated by p+1 similar "white fingers" of the complement of A.

Reason: For $z \rightarrow 0$, $z \in A$ is equivalent to $|R(z)e^{-z}| = |1 - e^{-z}(e_{p+1}z^{p+1} + e_{p+2}z^{p+2} + \dots)| > 1$

or $\operatorname{Re}(e_{p+1}z^{p+1} + (e_{p+2} - e_{p+1})z^{p+2} + \dots) < 0$.

Thus the sign of the error constant determines the color of the fingers.

Lemma 3. For $z \rightarrow \infty$, A tends to the negative half-plane with exactly two boundaries going to infinity.

Reason: elementary properties of e^z .

Lemma 4. Each bounded finger of A contains at least one pole of R ; each bounded white finger contains at least one zero. If several bounded fingers collapse, then they contain together at least as many poles or zeros as there are fingers.

This lemma is less trivial, see /1/.

While Lemmas 1 and 3 are linked to special properties of the exponential function, Lemmas 2 and 4 remain true for any order stars of two functions $R(z)$ and $S(z)$.

2. A-acceptability of Padé approximations.

For Padé approximations we have $p = \alpha + \omega$ and Lemmas 2 to 4 do not permit any other configuration as ω black fingers (bounded) to the right and α white bounded fingers to the left. Because of Lemma 1 and 4, we have A-acceptability iff the imaginary axis passes behind the last bounded black finger (as indicated by dots in Fig. 2). This is equivalent to

$$\alpha \leq \omega \leq \alpha + 2.$$

This includes the "conjecture of Ehle".

3. More results on rational approximations.

The following generalizes the Ehle-conjecture:

Theorem 5. If R is A-acceptable, $p \leq 2\alpha + 2$.

Proof. At least $[(p+1)/2]$ fingers of A start in \mathbb{C}^- . These have to be infinite and enclose at least $[(p+1)/2] - 1$ bounded white fingers. Since in total we have α zeros, it follows that $[(p+1)/2] - 1 \leq \alpha$. \square

The next theorem will below become the conjecture of Daniel and Moore:

Theorem 6. If R is A-acceptable, $p \leq 2\omega$.

Proof. At least $[(p+1)/2]$ fingers of A start in \mathbb{C}^+ . They cannot cross $i\mathbb{R}$ and must therefore be bounded. Since in total we have ω poles, it follows that $[(p+1)/2] \leq \omega$. \square

Theorem 7. If R is A-acceptable and $p = 2\omega$, then for the error constant $\operatorname{sign}(e_{p+1}) = (-1)^\omega$.

Proof. Look at the color of the fingers.

Theorem 8. If R is A-acceptable and $p = 2\omega$, then $|e_{p+1}| \geq \frac{\omega! \omega!}{(2\omega)!(2\omega+1)!}$.

Proof. Consider the order star B where $R(z)$ is "compared" to the diagonal approximation $R_{\omega\omega}(z)$.

The following are inverse results assuring A-acceptability:

Theorem 9. Suppose that $p \geq 2\omega - 1$ and $|R(\infty)| \leq 1$. Then R is A-acceptable.

Proof. The hypotheses need all poles in \mathbb{C}^+ . (see /1/).

Theorem 10. Suppose $p \geq 2\omega - 2$, $|R(\infty)| \leq 1$, and the coefficients of $Q(z)$ have alternating signs. Then R is A-acceptable.

A similar version of this has been proved differently by Crouzeix-Ruamps (1977). Theorems of Ehle's Thesis are special cases.

Theorem 11. Suppose $p \geq 2\omega - 3$, $|R(iy)| \leq 1$ for all y real, and the coefficients of $Q(z)$ have alternating signs. Then R is A-acceptable.

Proof. Similar as for 9 or 10. see /2/.

Following are the exact characterizations of rational approximations of order $2\omega - 3$ and $2\omega - 4$:

All rational approximations of order $2\omega - k$ can be written as

$$(5) \quad R(z) = \sum_{i=0}^{\omega} N^{(1)}(1) z^{\omega-1} / \sum_{i=0}^{\omega} N^{(1)}(0) z^{\omega-1}$$

where

$$N(t) = P_{\omega}(t) + A_1 P_{\omega-1}(t) + \dots + A_k P_{\omega-k}(t),$$

$$P_{\omega}(t) = \left(\frac{d}{dt}\right)^{\omega} (t^{\omega} (t-1)^{\omega}) \quad (\text{see /4/, Prop. 3}).$$

Theorem 12. $R(z)$ of (5) of order $2\omega - 3$ is A-acceptable iff

$$i) 1 + A_2 \geq 0, \quad ii) A_1 + A_3 \leq 0, \quad iii) A_3 \geq 0.$$

Theorem 13. $R(z)$ of (5) of order $2\omega - 4$ is A-acceptable iff

$$i) 1 + A_2 + A_4 \geq 0, \quad ii) A_1 + A_3 \leq 0, \\ iii) (2\omega - 5)A_1 A_4 + (2\omega - 1)A_3 \geq 0, \quad iv) A_1 \leq 0.$$

The proofs involve Theorems 9 and 11 above and some continuity arguments of the poles of $R(z)$ as functions of A_1, A_2, A_3, A_4 . See /6/ for details.

The following theorem gives an order bound for approximations with multiple poles or zeros:

Theorem 14. Suppose that R possesses only α_0 different zeros and ω_0 different poles. Then $p \leq \alpha_0 + \omega_0$.

Because of symmetry, real but different poles play sometimes the same role as one multiple pole. So this theorem is related to the highest attainable order with real poles. See the talk of S.P. Nørsett for details.

4. Extension to multi-step methods.

For a k -step method the rational function (1) is replaced by a characteristic polynomial

$$(1') \quad Q(z, R) = Q_0(z) R^k + Q_1(z) R^{k-1} + \dots + Q_k(z) = 0$$

$$Q_0(0) \neq 0, \quad \deg(Q_k) = \alpha, \quad \deg(Q_0) = \omega.$$

$R(z)$ becomes an algebraic function to be considered on its Riemann surface M . For linear multistep methods we have $\alpha \leq 1$, $\omega \leq 1$. For other classes of methods α and ω are related to the number of involved derivatives, stages etc.

From the fact that $R_1=1$ is a single zero of $Q(0,R)$ and can thus be analytically extended in a neighbourhood of zero to a holomorphic function $R_1(z)$, the order condition can now be written as

$$(2') \quad e^z - R_1(z) = e_{n+1} z^{n+1} + O(z^{n+2}).$$

The above Lemma 2 now carries over to the corresponding sheet of the Riemann surface, when we now consider the order star as a subset of M . Also for Lemmas 1,3, and 4 exist natural extensions. Thus:

Theorems 6,7,8, and 11 remain true in the multi-step case.

Theorems 6 and 8 have been known as the "conjecture of Daniel and Moore", the special case $\omega=1$ is Dahlquist's classical Theorem that $p=2$ is the highest attainable order of an A-stable linear multistep method.

Theorem 9 needs the following modification:

Theorem 9'. Suppose that $p \geq 2\omega-1$ and $|R(iy)| \leq 1$ for all $y \in \mathbb{R}$. Then R is A-acceptable.

This extends the result of Jeltsch (BIT 1978) to $\omega > 1$.

Theorems 5 and 14 do not extend as they are to the multistep case, since the proofs use the fact that \mathbb{C} is simply connected, which is no longer true for arbitrary Riemann surfaces.

5. Extension to second order equations.

If a stability test for methods for

$$y'' = f(t, y)$$

is made at $y'' + \beta y = 0$ with β real, the characteristic polynomial becomes

$$(1'') \quad Q(z, R) = Q_0(z^2)R^k + \dots + Q_k(z^2) = 0,$$

where now $R=1$ becomes a double zero of $Q(0,R)=0$. But since there appears z^2 in (1'') only, there is no monodromy around this point on the corresponding Riemann sheets and these two zeros continue as holomorphic functions in the neighbourhood of the origin. One of these two solutions then satisfies the order condition (2') and there is again a beautiful star on the corresponding sheet of the Riemann surface. So Theorems 6,7,8, and 11 remain true also in this case. The special case $\omega=1$ was proved by Dahlquist (BIT 1978). Theorem 6 has the following corollary:

Corollary 6'. If the solution of $Q(z, R)=0$ is P-stable (see Lambert-Watson, J. Inst. Math. Applics. 1976: this is equivalent with the fact that all roots of $Q(iy, R)=0$, y real, satisfy $|R| \leq 1$), then $p \leq 2\omega$.

See Hairer /3/ for details. Hairer has constructed unconditionally stable methods of higher order for this type of equations.

References.

- /1/ G.Wanner, E.Hairer, S.P.Nørsett, Order stars and stability theorems, BIT 18 (1978), 475-489
- /2/ G.Wanner, E.Hairer, S.P.Nørsett, When I-stability implies A-stability, BIT 18 (1978), p 503
- /3/ F.Hairer, Unconditionally stable methods for second order differential equations, to appear in Numer. Math., 1979
- /4/ S.P.Nørsett, G.Wanner, The real-pole sandwich for rational approximations and oscillation equations, appears in BIT 1979.
- /5/ S.P.Nørsett, G.Wanner, Perturbed collocation and Runge-Kutta methods, submitted to Numer. Math.
- /6/ K.Burrage, G.Wanner, Characterization of A-stable and B-stable Runge-Kutta methods of order $2s-3$ and $2s-4$, to be written.

S.P. NØRSETT
INSTITUTE OF NUMERICAL MATHEMATICS
UNIVERSITY OF TRONDHEIM, TRONDHEIM-NTH

The collocation method for obtaining an approximate solution to the initial value problem

$$(1) \quad y' = f(t, y), \quad t \in (a, b), \quad y(a) = y_0,$$

amounts to finding a polynomial u of degree at most m such that

$$i) \quad u(t_0) = y_0$$

$$ii) \quad u'(t_0 + c_i h) = f(t_0 + c_i h, u(t_0 + c_i h)),$$

$$i = 1, \dots, m,$$

$$iii) \quad y_1 = u(t_1 = t_0 + h)$$

where c_i , $i = 1, \dots, m$ are m distinct real numbers. By using a result of Alekseev and Gröbner on the nonlinear Variation-of-constants formula for (1) a very short and nice proof can be given for the already known result that the collocation method possesses the same order as the corresponding quadrature formulae.

When the collocation scheme is applied to the test equation $y' = \lambda y$, $z = \lambda h$, we get

$$y_1 = R(z)y_0 = (P(z)/Q(z))y_0$$

where

$$P(z) = N(1)z^m + \dots + N^{(m)}(1),$$

$$Q(z) = N(0)z^m + \dots + N^{(m)}(0)$$

and $N(t) = (t - c_1) \dots (t - c_m)/m!$. $R(z)$ is an approximation to $\exp(z)$ of order $s \geq m$ if the method has order s . (N is called

the N -polynomial to R and is related to the C -polynomial p of Nørsett by $N(t) = (-1)^m p(1-t)$.)

It is well known that the collocation method is equivalent to a special subclass of Runge-Kutta methods with RK-matrix

$$A = \left\{ a_{ij} = \int_0^{c_i} \ell_j(\tau) d\tau \right\}_{i,j=1}^m \quad \text{where}$$

$$\ell_j(\tau) = \prod_{k \neq j} (\tau - c_k) / \prod_{k \neq j} (c_j - c_k).$$

By writing $Q(z) = (1 - \gamma_1 z) \dots (1 - \gamma_m z)$ we have that $\gamma_1, \dots, \gamma_m$ are the eigenvalues of A and that if γ_j is a simple eigenvalue of A , a corresponding eigenvector x_j of A is

$$x_j = (x_{1j}, \dots, x_{mj})^T$$

where

$$x_{ij} = N^{(i)}(c_j) \gamma_j + \dots + N^{(m)}(c_j) \gamma_j^m.$$

These results are, as pointed out by Butcher, essential in connection with the practical implementations of implicit RK-methods. In particular we are interested in RK-methods where the eigenvalues are real and hopefully equal. Two questions arise: first which collocation points do we need in such a case and secondly what is the maximum order of the corresponding method.

An upper bound on the order can be given by considering the order of rational approximations $R(z)$ to $\exp(z)$ where R has real poles. For example, we can show by using the theory of *order stars* developed

by Wanner, Hairer and Nørsett: the following interesting result :

Let $R(z) = P_k(z)/Q_j(z)$, $P_k \in \pi_k$, $Q_j \in \pi_j$

where Q_j has $r \leq j$ complex different zeros. Then the order s satisfies

$$s \leq \begin{cases} k+r & \text{if } Q_j \text{ has no real zero} \\ k+r+1 & \text{if } Q_j \text{ has at least one real zero.} \end{cases}$$

This result was earlier obtained for the case $r = 0$ by Nørsett & Wolfbrandt in a rather more technical way. The proof is now short and is a counting of bounded dual fingers versus unbounded dual fingers.

The A-stability of these methods, and also of all one-step methods giving $y_1 = R(z)y_0$ when applied to the test equation $y' = \lambda y$, reduces to the question of A-acceptability of $R(z) = P_k(z)/Q_j(z)$, i.e. $|R(z)| < 1$ for $z \in \mathbf{C}^-$. For the optimal order approximation, $s = k+j$, R is an entry in the Padé-table. By introducing the concept of *order stars* (see the talk by G. Wanner) one easily proves that the diagonal and the first two sub-diagonal entries are the *only* A-acceptable approximations.

When Q_j has only real zeros $\gamma_j = \gamma$, $j = 1, \dots, k$, $k = j$, the approximations have the form (of order at least k)

$$R(z) = \left(\sum_{m=0}^k (-1)^m L_k^{(k-m)} (1/\gamma) (\gamma z)^m \right) / (1-\gamma z)^k,$$

$$\gamma \in \mathbf{R}.$$

Optimal order $k+1$ is obtained for γ a solution to $L_{k+1}'(1/\gamma) = 0$. Let the k different γ -values be denoted by γ_i with, $0 < 1/\gamma_1 < 1/\gamma_2 < \dots < 1/\gamma_k$. For each of these γ -values we obtain the restricted-Padé-approximations, and an interesting property is that the order star corresponding to $\gamma = \gamma_v$ contains just *one* bounded finger of multiplicity $k-v+1$. Based on that result the A-acceptability of the restricted-Padé-

approximations is as the following table shows

k	1	2	3	5
v	1	1	1	2

When γ is a solution to $L_k(1/\gamma) = 0$, $Q_k \in \pi_{k-1}$ and L-acceptable approximations may appear. Some results in that direction will be discussed as well as the I-acceptability of the special approximations

$$R(z) = P_{2k}(z)/(1-\gamma^2 z^2)^k$$

suitable in connection with integrating discretized hyperbolic equations. Some of the questions raised by Baker and Bramble who first introduced these functions will be dealt with.

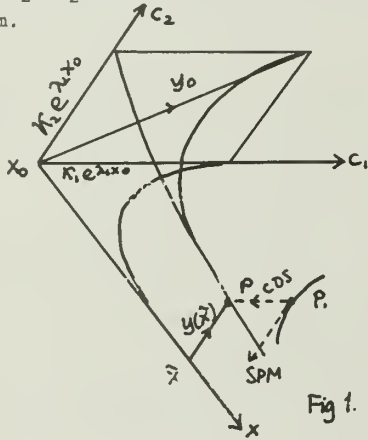
For further details see the following papers,

- [1] G. Wanner, E. Hairer, S.P. Nørsett; "Order stars and stability theorems", BIT 18 (1978), 475-489.
- [2] S.P. Nørsett, G. Wanner; "The real-pole sandwich for rational approximations and oscillation equations". To appear BIT.
- [3] S.P. Nørsett, G. Wanner; "Perturbed collocation and Runge-Kutta methods". Submitted to Num. math.

J D Lambert
University of Dundee

§1 Introduction

The solution of the 2×2 linear constant coefficient system $y' = Ay$, $y(x_0) = y_0$, where A has real eigenvalues $\lambda_1 \ll \lambda_2 < 0$ and eigenvectors c_1, c_2 is $y(x) = \kappa_1(\exp(\lambda_1 x))c_1 + \kappa_2(\exp(\lambda_2 x))c_2$ represented in the following diagram.



$P_1[\tilde{x}, (\kappa_1 + \epsilon_1)(\exp(\lambda_1 \tilde{x}))c_1 + (\kappa_2 + \epsilon_2)(\exp(\lambda_2 \tilde{x}))c_2]$ is a numerical approximation to the exact solution point P . Even if P_1 is close to P , the high gradient at P_1 invokes instability if an explicit method is used. Correcting P_1 in the c_1 -direction was proposed by Lambert [1] and developed by Alföld and Lambert [2] into the CDS technique (Correction in the Dominant Space).

The new proposal is to correct P_1 in the direction of the tangent to the local integral curve through P_1 . For all κ_1 , this tangent passes through $S[\tilde{x} - 1/\lambda_1, (\kappa_2 + \epsilon_2)(1 - \lambda_2/\lambda_1)(\exp(\lambda_2 \tilde{x}))c_2]$, which we call the safe point, since it invokes no high gradients.

Consider the $m \times m$ linear system $y' = Ay + g(x)$, where A has eigenvectors $\{c_t\}$ and eigenvalues $\{\lambda_t\}$ such that

$$\lambda_1 \text{ real, } \lambda_1 \ll \min_{t=2, \dots, m} \operatorname{Re} \lambda_t < 0 \quad (1.1)$$

If (1.1) is satisfied we say that the system, and the matrix A , are simply separably stiff. The exact solution at $x = \tilde{x}$ is represented by the point $P[\tilde{x}, \kappa_1(\exp(\lambda_1 \tilde{x}))c_1 + \psi(\tilde{x})]$, and the numerical solution by $P_1[\tilde{x}, (\kappa_1 + \epsilon_1)(\exp(\lambda_1 \tilde{x}))c_1 + \psi(\tilde{x}) + \delta(\tilde{x})]$. The tangent at P_1 to the local integral curve now contains the safe point $S[\tilde{x} - 1/\lambda_1, \psi(\tilde{x}) + \delta(\tilde{x}) - (\psi'(\tilde{x}) + \delta'(\tilde{x}))/\lambda_1]$, which, note, no longer lies in the subdominant space. Since $1/\lambda_1$ is small we could accept the solution at S as an approximation to $y(\tilde{x})$; this invokes a persistent error (cf CDS) of $\eta'(\tilde{x})/\lambda_1$. More advantageously, we can accept the solution at S as an approximation to $y(\tilde{x} - 1/\lambda_1)$, which invokes a persistent error of $\eta''(\tilde{x})/\lambda_1^2$.

§2 One-step Safe Point Methods

Problem

$y' = A(x)y + g(x) =: f(x, y)$, $y(x_0) = y_0$,
 $y, g \in \mathbb{R}^m$, eigenvalues $\{\lambda_t(x)\}$ satisfying (1.1)
 $\forall x \in [x_0, x_N]$.

Procedure

- (1) Starting from x_n , take forward step, length h_n , by conventional explicit method, the basic method.
- (2) Compute λ_1 by the power method. (Note (i) the stiffer the better! (ii) estimates for initial iterate available from previous step; (iii) eigenvectors not needed (cf CDS).)
- (3) Proceed to safe point, and accept as an approximation to $y(x_n + h_n + \xi_{n+1})$, where $\xi_{n+1} = -1/\lambda_1(x_n + h_n)$.

This gives an irregular grid of data points, hence the motivation for one-step basic methods, which will yield numerical solutions on the grid

$$\{x_n | x_n = x_0 + \sum_{j=0}^{n-1} (h_j + \xi_{j+1})\}.$$

Algorithm (SPM)₁

$$\left. \begin{aligned} z_{n+1} &= y_n + h_n \phi_f(x_n, y_n; h_n) & (i) \\ \xi_{n+1} &= -1/\lambda_1 (x_n + h_n) & (ii) \\ y_{n+1} &= z_{n+1} + \xi_{n+1} f(x_n + h_n, z_{n+1}) & (iii) \end{aligned} \right\} (2.1)$$

Local TE = $O(h_n^{p+1}) + O(\xi_{n+1}^2)$,
p the order of (i)

Note: It is of course possible to start with the correction step (iii); this is equivalent to applying (2.1) to a perturbed form of the problem.

§3 Other Safe Points

Regard (2.1(iii)) as an Euler step with ξ_{n+1} as discretization parameter (cf singular perturbation?). We could replace this by RK_q (any q-stage explicit Runge-Kutta method of order q, q ≤ 4). Clearly, from Fig 1, S can no longer be the safe point. It turns out that RK₂ has no safe point, but RK₃ has a safe point given by $\xi = \gamma/\lambda_1$, $\gamma = -1.596071638$. Using this in place of (ii) and replacing (iii) by an RK₃ method, (2.1) yields a method, which we label (SPM)₁³, with local TE = $O(h_n^{p+1}) + O(\xi_{n+1}^4)$; the persistent error can now be safely ignored, even for systems of modest stiffness. In the sequel we consider only correction steps of the form given in (2.1), but in every case replacement by a correction based on RK₃ is possible.

§4 Stability and Sensitivity of (SPM)₁

Definition

Let A be simply separably stiff and have distinct eigenvalues $\{\lambda_t | t = 1, 2, \dots, m\}$. Then a SPM is said to be dominantly stable if all solutions of the difference equation in $\{y_n\}$ which result from applying the method to $y' = Ay$ tend to zero as $n \rightarrow \infty$ for all step lengths h_n such that $h_n \lambda_t \in \mathcal{L}_B$, $t = 2, 3, \dots, m$, where \mathcal{L}_B is the region of absolute stability of the basic method.

Theorem

(SPM)₁ and (SPM)₁³ are dominantly stable.

Sensitivity

Applying (2.1(i)) to $y' = Ay$ yields $z_{n+1} = P(h_n A)y_n$, where $P(\cdot)$ is a polynomial of degree p. Dominant stability depends on the fact that $(1 + \xi \lambda_t)P(h_n \lambda_t) = 0$ when $t = 1$. If we denote by $\tilde{\lambda}_1$ the value of λ_1 computed by the power method, then stability is retained only if $|(1 - \lambda_1/\tilde{\lambda}_1)P(h_n \lambda_1)| < 1$, which leads to the requirement $(\tilde{\lambda}_1 - \lambda_1)/\tilde{\lambda}_1 \sim (h \lambda_1)^{-p}$, an excessive

precision. Analogous difficulties beset CDS when the basic method is one-step.

§5 Linear Multistep SPM

In order to obtain a solution on a regular grid, we propose to correct by one forward (Euler) step to the safe point and then back by a further (Euler) step. This process retains the same safe point (eg Fig 1). We now use as basic method the K-step explicit LMM (ρ^*, σ^*) .

Algorithm (SPM)_k

$$\left. \begin{aligned} z_{n+k} &= (E^k - \rho^*(E))y_n + h\sigma^*(E)f(x_n, y_n) & (i) \\ \xi_{n+k} &= -1/\lambda_1 (x_{n+k}) & (ii) \\ \hat{y}_{n+k} &= z_{n+k} + \xi_{n+k} f(x_{n+k}, z_{n+k}) & (iii) \\ y_{n+k} &= \hat{y}_{n+k} - \xi_{n+k} f(x_{n+k} + \xi_{n+k}, \hat{y}_{n+k}) & (iv) \end{aligned} \right\} (5.1)$$

Local TE = $O(h^{p+1}) + O(\xi_{n+k}^2)$

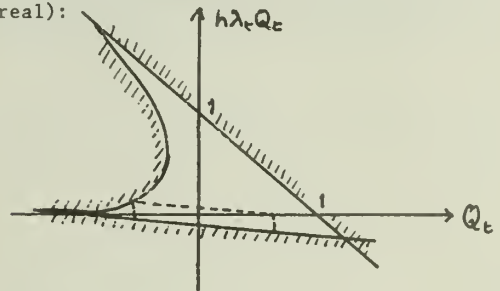
(SPM)_k³ is similarly defined, replacing (iii) and (iv) by RK₃ steps. (SPM)_k is absolutely stable if the polynomial

$$\pi(r; h\lambda_t, \xi\lambda_t) := r^k - Q_t[r^k - \rho^*(r) + h\lambda_t \sigma^*(r)] \quad (5.2)$$

is Schur for $t = 1, 2, \dots, m$, where

$Q_t = 1 - (\xi\lambda_t)^2$. This is clearly so for $t = 1$, while for $t = 2, \dots, m$, $\pi = \rho^* - h\lambda_t \sigma^* + O(\lambda_t/\lambda_1)^2$ and we expect a stability region little different from that of (ρ^*, σ^*) .

For $(\rho^*, \sigma^*) = 4$ th order Adams-Bashforth (5.2) is Schur in the following region (where we assume λ_t real):



It follows that for $t = 2, \dots, m$ the interval of absolute stability is only slightly less than that of AB₄. Let $t = 1$ in diagram, and for Q_1 read $1 - (\lambda_1/\tilde{\lambda}_1)^2$. Then stability is retained if $(Q_1, h\lambda_1 Q_1)$ lies in the indicated parallelogram, which implies the much improved sensitivity requirement $h|\lambda_1 - \tilde{\lambda}_1| < .077$.

§6 Predictor-corrector SPM

A predictor (ρ^*, σ^*) and corrector (ρ, σ) in PECE mode usually has better absolute stability than (ρ^*, σ^*) . Applying the correction process (5.1(iii)(iv)) after the PECE cycle - indicated by PECEξ - raises the sensitivity problems encountered

with $(SPM)_1$. However, for the mode $PE\xi CE\xi$, (5.2) is (5.2) is replaced by

$$\pi = (1-Q_t)(1+h\lambda_t\beta_k Q_t)r^k + Q_t[\rho-h\lambda_t\sigma+h\lambda_t\beta_k Q_t(\rho^*-h\lambda_t\sigma^*)] .$$

An order of magnitude argument suggests a good compromise between sensitivity and subdominant stability is got by setting to zero terms in $h\lambda_t Q_t$, which forces the choice $\sigma^* = \beta_k r^k$, ie (ρ, σ) is a backward differentiation formula (BDF). Further, it is found to be advantageous to choose $\rho^* = \rho$. The PL pair is then completely specified by

$$(\rho, \sigma) = \text{BDF of order } k ;$$

$$\rho^* = \rho ; (\rho^*, \sigma^*) \text{ of order } k .$$

§7 Extensions

The extension to the general separably stiff linear problem (ie more than one dominant eigenvalue) is straightforward; but computing estimates for the dominant eigenvalues becomes more difficult. Nonlinear systems may also be tackled by evaluating $\lambda_1 = \lambda_1(x, y)$, dominant eigenvalue of the Jacobian, at appropriate approximate values for y . Success will depend on the eigensystem of the Jacobian varying slowly with x .

References

- [1] Lambert, J D "The numerical integration of a special class of stiff differential systems", Proc 5th Manitoba Conf on Num Math, Oct 1975.
- [2] Alfeld, P and Lambert, J D "Correction in the dominant space: a numerical technique for a certain class of stiff initial value problems", Math Comp Vol 31, No 140, 922-938, Oct 1977.

THE NEED FOR IMPROVED NUMERICAL ALGORITHMS IN MOLECULAR SCATTERING*

Lowell D. Thomas
 National Resource for Computation in Chemistry
 Lawrence Berkeley Laboratory
 University of California
 Berkeley, California 94720 USA

Many problems in theoretical chemistry require an understanding at the microscopic level of molecular scattering. That is, an understanding of what internal and external changes take place as two molecules approach one another, collide and then recede from one another. From this knowledge at the microscopic level one can then calculate many properties of macroscopic chemical systems which cannot be measured presently in the laboratory nor observed directly in our environment.

The applications of molecular scattering information are many and far-reaching. In laser physics this information is needed in the search for new chemical systems with better lasing properties and for performance improvement of existing ones. The discovery within the past ten years of a rich variety of molecules and molecular clouds in interstellar space has created a need in astrophysics for molecular scattering data. Many interstellar molecules cannot be made in earth-bound laboratories, leaving computation as the only method for the determination of their properties. The study of our own atmosphere involves many molecular scattering processes, including, for example, the formation and destruction of ions in the upper atmosphere and the creation of air pollutants. Many energy-related problems also require understanding of molecular scattering events. To understand combustion processes requires knowledge of the associated reactive scattering events and in nuclear reactors there are many inelastic and reactive collisions involved in the final disposition of the generated energy. This listing is just a brief mention of a few of the applications.

Quantum mechanical approaches to molecular scattering can be reduced to the solution of a system of second-order, ordinary differential equations of the form,

$$y''(r) + k^2 y(r) = V(r)y(r) \quad (1)$$

$$k^2 > 0 \quad (2)$$

$$y(0) = 0 \quad (3)$$

$$\lim_{r \rightarrow \infty} y(r) = \sin(kr + \eta) + \cos(kr + \eta)R \quad (4)$$

$$\eta = \text{constant} \quad (5)$$

Here $V(r)$ is called the potential energy function or most often simply the potential. Another important class of equations have non-local potentials, i.e.,

$$y''(r) + k^2 y(r) = \int_0^\infty V(r, r') y(r') dr' \quad (6)$$

However, the discussion here will be limited to the local case. Eq. (1) will generally be a matrix equation with $V(r)$ a real, symmetric matrix and k^2 diagonal. In this case the sin and cos functions in Eq. (4) are to be regarded as diagonal matrices and R is real symmetric. The matrix R is the desired quantity. From it all micro- and macroscopic scattering information can be derived.

This problem is most commonly solved by starting with Eq. (3), assuming an arbitrary, non-singular matrix for $y'(0)$ and then numerically integrating outward to some large r such that $V(r) \approx 0$. the numerical solution then has the form

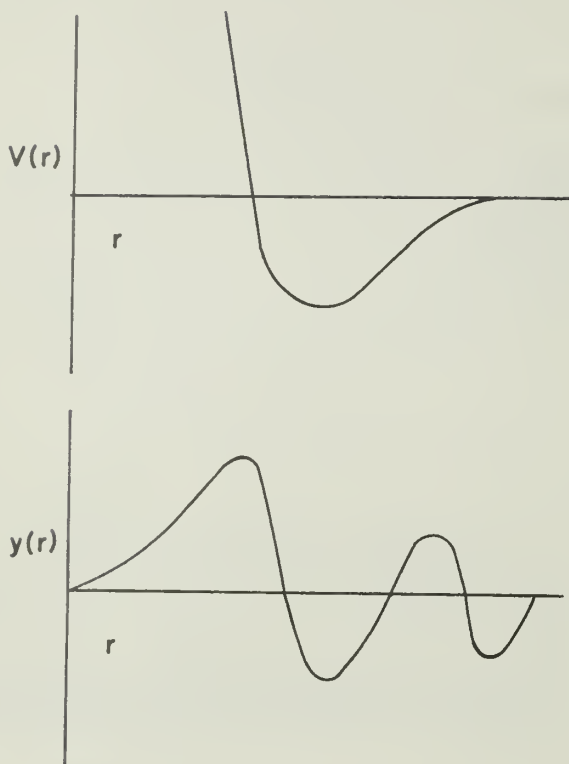
$$y(r) = \sin(kr + \eta)A + \cos(kr + \eta)B \quad (7)$$

and the desired solution is found by multiplying from the right by A^{-1} . That is

$$R = BA^{-1} \quad (8)$$

The single equation problem illustrated in the figure below is representative of the nature of the problem. For small r where $V(r) > k^2$, the solution has exponential behavior. For large r where $V(r) \ll k^2$ the solution is oscillatory. Various methods have been devised to compute these functions, some of which try to take advantage of their known limiting behavior. A brief survey of these methods with representative references follows.

*This research was supported in part by the National Resource for Computation in Chemistry under a grant from the National Science Foundation and the Basic Energy Sciences Division of the United States Department of Energy under Contract No. W-7405-ENG-48.



The Numerov¹ and de Vogelaere² algorithms are straight forward and have found wide application. The method of Gordon³ which makes a piecewise linear approximation to $V(r)$ is especially advantageous in the large r region. The variable phase method⁴ represents y as a product of an oscillator y function and a modulating function which is smooth and slowly varying over a large part of the integrating range. Several methods^{5,6} have also been developed for integrating the integral equations equivalent to Eq. (1). In addition a number of unconventional methods have recently been developed. These include the L^2 methods⁷ (the solutions y are expanded in a basis of square integrable functions), the R-matrix propagator method⁸ and an iterative method for finding a single column of the solution matrix y .⁹ It is also possible to formulate the problem as a smaller system of linear, second-order partial differential equations and finite element methods for solving these have recently been proposed.¹⁰

Because of the matrix multiplication in Eq. (1), the computing time required for most of these methods increases as N^3 where N is the order of the matrix. The maximum cases solved to date typically involve $N=100$. The greatest need at the moment in molecular scattering, is for methods which allow a significant increase in N .

Other important needs for improved algorithms are 1) faster, more efficient programs, 2) error estimates with variable and automatic step size selection, and 3) programs which are generally reliable and easy to use.

With the exception of Gordon's method³ computer programs for these methods are not generally available to or usable by people other than the developers. At the National Resource for Computation in Chemistry (NRCC), we are now attempting to bring together, test and make available the best of these programs. At this meeting, we hope to stimulate a broader dialogue on such problems between theoretical chemists and the community of numerical mathematicians.

References

LBL-8882

1. A. C. Allison, J. Comput. Phys. 6, 378 (1970).
2. W. A. Lester, Jr., in Methods in Computational Physics, Vol. 10, pg. 211, ed. B. Alder, S. Fernbach and M. Rotenberg (Academic Press, New York 1971).
3. R. G. Gordon, J. Chem. Phys. 51, 14 (1969).
4. M. LeDourneuf and Vo Ky Lan, J. Phys. B. 10, L35 (1977).
5. N. W. Sams and D. J. Kouri, J. Chem. Phys. 51, 4809 (1969).
6. D. Secrest, in Methods in Computational Physics, Vol. 10, pg. 243, ed. B. Alder, S. Fernbach and M. Rotenberg (Academic Press, New York 1971).
7. D. A. Levin, T. N. Rescigno and V. McKoy, Phys. Rev. A 16, 157 (1977).
8. E. B. Stechel, R. B. Walker and J. C. Light, J. Chem. Phys. 69, 3518 (1978).
9. L. D. Thomas, J. Chem. Phys. 70, 000 (1979).
10. H. Rabitz, A. Askar and A. S. Cakmak, Chem. Phys. 29, 61 (1978).

FACTORED, A-STABLE, LINEAR MULTISTEP METHODS - AN ALTERNATIVE

TO THE METHOD OF LINES FOR MULTIDIMENSIONS

R. F. Warming and Richard M. Beam

Computational Fluid Dynamics Branch

Ames Research Center, NASA

Moffett Field, California 94035

INTRODUCTION. Historically, the development and analysis of methods for ordinary differential equations (ODEs) have been more advanced than those for partial differential equations (PDEs). The present state of numerical methods is no exception; therefore, it behooves the numerical analyst to exploit sophisticated ODE methods for the numerical solution of PDEs.

The method of lines, for example, capitalizes on reliable ODE packages to solve PDEs. However, as the number of spatial dimensions increases, the direct application of the method of lines produces excessively large systems of ODEs. As an illustration, consider the two-dimensional hyperbolic system

$$(1) \quad \frac{\partial u}{\partial t} + \frac{\partial F(u)}{\partial x} + \frac{\partial G(u)}{\partial y} = 0,$$

where u is an m -component vector. For the equations of gas dynamics $m = 4$ and if the spatial derivatives in (1) are replaced by spatial differences and the computational mesh is chosen with 50 grid points in each spatial direction, the end result is a system of 10^4 ODEs. While this system might not overwhelm the more robust ODE packages, the three-dimensional counterpart for practical physical problems would produce a system of order 10^6 stiff nonlinear ODEs - a formidable challenge for any ODE package. The stiff character of the system implies that an efficient solution algorithm must be implicit, hence we limit our discussion to implicit methods. Clearly, the direct application of the method of lines to multidimensional PDEs does not take advantage of the sparse structure of the ODE system - unless the ODE package is appropriately designed [7].

In the development of implicit numerical methods for PDEs, the greatest improvement in numerical efficiency was the introduction of alternating direction implicit (ADI) methods [5,6,10], which reduce multidimensional computational problems to a sequence of simpler one-dimensional problems. In the context of the method of lines, ADI methods are analogous to an approximate factorization of the large matrix which must be "inverted" by the stiff ODE package. Although in principle the approximate factorization could be designed into the ODE package, in practice it would be difficult without an *a priori* knowledge of the PDE system. A more efficient technique is to discard the direct application of the method of lines and introduce the ODE methods prior to the spatial differencing.

The purpose of this paper is to demonstrate that the methods and analyses from the numerical ODE development can be directly applied to the develop-

ment of efficient PDE algorithms. We choose the linear multistep methods (LMMs) and focus on those which are A-stable. As a vehicle for the demonstration we outline the development of an A-stable ADI-LMM for a mixed hyperbolic-parabolic system. We do not propose the direct application of existing ODE packages; however, it is hoped that many features of these packages (variable time step, error control, etc.) can be implemented in PDE packages.

PRELIMINARIES. Consider a nonlinear mixed hyperbolic-parabolic system of the form

$$(2) \quad \frac{\partial u}{\partial t} + \frac{\partial F(u)}{\partial x} + \frac{\partial G(u)}{\partial y} = \frac{\partial V(u, u_x, u_y)}{\partial x} + \frac{\partial W(u, u_x, u_y)}{\partial y},$$

where u is an unknown m -component vector, and F and G are given (in general, nonlinear) vector functions of u . The vectors V and W are vector functions not only of the components of u but also of $\partial u/\partial x$ and $\partial u/\partial y$.

Physical problems associated with equations of form (2) (e.g., the compressible Navier-Stokes equations) are typically stiff by virtue of disparate characteristic speeds and/or length scales. In addition, the parabolic terms on the right-hand side of (2) are often negligible except in thin layers of the solution domain. As a result, the eigenvalues associated with a locally linearized version of (2) are widely distributed with large imaginary parts. Consequently, A-stable integration formulas provide ideal time-differencing approximations.

As is well known, the order of accuracy of an A-stable linear multistep method cannot exceed two [2]. At present, the application of higher order A-stable methods such as multistep multiderivative methods appears to be impractical for PDEs as complex as (2). However, there exists a large class of physical problems where second-order temporal accuracy suffices. For the equations of gas dynamics at high Reynolds number, solutions of (2) exhibit large spatial gradients in localized regions such as shock waves, shear layers, boundary layers, etc. In addition, the fastest characteristic speeds are often of only marginal importance. For such problems, numerical schemes which combine fourth-order spatial accuracy with second-order temporal accuracy are often considered "optimal" in terms of both storage requirements and computational effort (work). One must also keep in mind that (2) is an initial boundary value problem; consequently, it is very difficult to supply the proper boundary conditions (for a well-posed

problem) at the advanced time level when higher order temporal methods are used.

We have recently shown [13] that A-stable LMMs combined with approximate factorization provide a natural framework for the construction of unconditionally stable ADI methods for PDEs. The main emphasis of [13] was linear stability theory and we considered only model equations. In the following section we outline the extension of [13] to non-linear PDEs.

ALGORITHM DEVELOPMENT. A linear k-step method for

$$(3) \quad \frac{du}{dt} = f(u), \quad u(0) = u_0$$

can be written as

$$(4) \quad \rho(E)u^n = \Delta t \sigma(E)f^n,$$

where

$$(5) \quad \rho(\zeta) = \sum_{j=0}^k \alpha_j \zeta^j, \quad \sigma(\zeta) = \sum_{j=0}^k \beta_j \zeta^j,$$

E is the shift operator, $Eu^n = u^{n+1}$, and Δt is the time increment. Henceforth, we consider only A-stable LMMs; i.e., we assume (4) is A-stable.

In the procedure suggested in [13] for constructing ADI schemes, the implicit operator to be "inverted" is constructed so that the unknown variable to be determined at each time step is $\rho(E)u^n$ rather than u^{n+k} . Hence, as a precursor to formulating an ADI scheme, we rewrite (4) as

$$(6) \quad \rho(E)u^n - \omega \Delta t \rho(E)f^n = \Delta t [\sigma(E) - \omega \rho(E)]f^n$$

where $\omega = \beta_k/\alpha_k$. The parameter ω is defined so that the operator $\sigma(E) - \omega \rho(E)$ on the right-hand side of (6) is at least one degree lower than the operator $\rho(E)$ on the left-hand side. Hence the right-hand side of (6) can be computed explicitly, i.e., from known data when advancing the numerical solution from $n+k-1$ to $n+k$. The restriction to second-order temporal accuracy allows several approximations to be made which greatly simplify the final algorithm. We illustrate these approximations for the LMM (6).

(a) Local linearization. Since (6) is a nonlinear system of difference equations, an iterative solution procedure such as the Newton's method could be used at each time step. For schemes whose order of accuracy is ≤ 2 , iterative solutions can be avoided by a local linearization. By using a local Taylor series and the consistency condition for (4) it is easy to show that

$$(7) \quad \rho(E)f^n = J^{n+p} \rho(E)u^n + O(\Delta t^2),$$

where J is the Jacobian matrix $J = \partial f/\partial u$ and p is an integer in the range $0 \leq p < k$. With this approximation, (6) becomes

$$(8) \quad (I - \omega \Delta t J^{n+p}) \rho(E)u^n = \Delta t [\sigma(E) - \omega \rho(E)]f^n$$

which is a second-order-accurate linear equation for $\rho(E)u^n$.

(b) Quasi-one-leg method. The evaluation of the right-hand side of (8) can be reduced to a single function evaluation by noting that

$$[\sigma(E) - \omega \rho(E)]f^n = f(\bar{u}) + O(\Delta t^2),$$

where

$$\bar{u} = [\sigma(E) - \omega \rho(E)]u^n.$$

This is a variant of one-leg methods [3] corresponding to the LMM (4). Hence (8) becomes

$$(9) \quad (I - \omega \Delta t J^{n+p}) \rho(E)u^n = \Delta t f(\bar{u}) + O(\Delta t^3).$$

Example: The linear one-step method $\rho(E) = E - 1$, $\sigma(E) = \theta E + (1 - \theta)$, $\sigma(E) - \omega \rho(E) = 1$, is A-stable for $\theta \geq 1/2$ and is second-order accurate when $\theta = 1/2$. For this example, algorithm (9) becomes

$$(I - \theta \Delta t J^n)(u^{n+1} - u^n) = \Delta t f(u^n).$$

This noniterative ODE method was suggested independently by Lomax [9] and Liniger and Willoughby [8]. This scheme is equivalent to solving (4) by Newton's method and stopping the iteration after one step.

(c) Approximate factorization. For PDE applications it is convenient to split (3) as

$$(10) \quad \frac{du}{dt} = f(u) = f_1(u) + f_2(u).$$

The Jacobian is then

$$(11) \quad J = \frac{\partial f}{\partial u} = \frac{\partial f_1}{\partial u} + \frac{\partial f_2}{\partial u} = J_1 + J_2,$$

and the algorithm (9) becomes

$$(12) \quad [I - \omega \Delta t (J_1^{n+p} + J_2^{n+p})] \rho(E)u^n = \Delta t f(\bar{u}).$$

An approximate factorization of (12) yields

$$(13) \quad (I - \omega \Delta t J_1^{n+p})(I - \omega \Delta t J_2^{n+p}) \rho(E)u^n = \Delta t f(\bar{u}).$$

It is shown in [13] that this approximate factorization does not upset either the second-order temporal accuracy, or the unconditional stability of the method when applied to linear test equation $du/dt = \lambda_1 u + \lambda_2 u$ ($\text{Re} \lambda_1, \lambda_2 \leq 0$). The A-stability is preserved because we have chosen $\rho(E)u^n$ as the unknown variable.

For simplicity, we illustrate the PDE application by considering only the hyperbolic equation (1). The algorithm for the complete equation (2) will be given in the final manuscript. Comparing (10) and (1), we identify the differential operators

$$(14) \quad f_1(u) = -\frac{\partial F(u)}{\partial x}, \quad f_2(u) = -\frac{\partial G(u)}{\partial y}$$

The Jacobians (11) are replaced by the operators

$$(15) \quad J_1 \leftarrow -\frac{\partial}{\partial x} A, \quad J_2 \leftarrow -\frac{\partial}{\partial y} B,$$

where $A = \partial F/\partial u$, $B = \partial G/\partial u$ are Jacobian matrices. (For the equations of gas dynamics in two spatial dimensions, these are 4×4 matrices.) Inserting (14) and (15) in (13) yields*

$$(16) \quad \left(I + \omega \Delta t \frac{\partial}{\partial x} A^{n+p} \right) \left(I + \omega \Delta t \frac{\partial}{\partial y} B^{n+p} \right) \rho(E)u^n = -\Delta t \left[\frac{\partial F(\bar{u})}{\partial x} + \frac{\partial G(\bar{u})}{\partial y} \right].$$

An obvious computational sequence to implement (16) as an ADI method is

*In (16), notation of the form

$$(I + \omega \Delta t \frac{\partial}{\partial y} B) \rho(E)u^n$$

denotes

$$\rho(E)u^n + \omega \Delta t \frac{\partial}{\partial y} (B \rho(E)u^n)$$

$$(17a) \quad \left(I + \omega \Delta t \frac{\partial}{\partial x} A^{n+p} \right) \rho(E) u^* = -\Delta t \left[\frac{\partial F(\bar{u})}{\partial x} + \frac{\partial G(\bar{u})}{\partial y} \right],$$

$$(17b) \quad \left(I + \omega \Delta t \frac{\partial}{\partial y} B^{n+p} \right) \rho(E) u^n = \rho(E) u^*,$$

$$(17c) \quad u^{n+k} = \frac{1}{\alpha_k} \left[\rho(E) u^n - \sum_{j=0}^{k-1} \alpha_j E^j u^n \right],$$

where $\rho(E)u^*$ is a dummy temporal difference. The spatial derivatives appearing in (17) are to be approximated by appropriate finite-difference quotients. For three-point spatial difference approximations, the x - and y -operators on the left side of (17a,b) each require the solution of a block-tridiagonal system of equations with each block having dimensions $m \times m$, where m is the number of components of the vector u . There is an obvious extension of (17) to three-spatial dimensions.

In practice, two-step A-stable methods are of primary interest. The class of all two-step methods which are at least second-order accurate can be written as

$$(18) \quad (1 + \xi)u^{n+2} - (1 + 2\xi)u^{n+1} + \xi u^n = \Delta t \left[\theta f^{n+2} + \left(\xi - 2\theta + \frac{3}{2} \right) f^{n+1} - \left(\xi - \theta + \frac{1}{2} \right) f^n \right],$$

where θ, ξ are arbitrary real numbers. By applying the theory of positive real functions [4], one finds that (18) is A-stable if and only if

$$\xi \leq 2\theta - 1, \quad \xi \geq -\frac{1}{2}.$$

MIXED SPATIAL DERIVATIVES. The appearance of mixed spatial derivatives $\partial^2/\partial x \partial y$ on the right-hand side of (2) precludes an efficient reduction of a multidimensional problem into a product of one-dimensional operators. This difficulty can be avoided by treating the mixed derivatives with an explicit LMM which is "compatible" with an A-stable LMM. We demonstrate the rather remarkable result that this can be done and still retain unconditional stability. The only penalty is that the parameter space for A-stability is slightly reduced.

NUMERICAL COMPUTATIONS. Finally, we give a brief survey [1,11,12] of recent two- and three-dimensional flow computations made using ADI-LMMs of the form (17).

REFERENCES

1. R. M. Beam and R. F. Warming, An implicit factored scheme for the compressible Navier-Stokes equations, AIAA J. 16 (1978), 393-402.
2. G. Dahlquist, A special stability problem for linear multistep methods, BIT 3 (1963), 27-43.
3. G. Dahlquist, Error analysis for a class of methods for stiff non-linear initial value problems, Lecture Notes in Mathematics 506, Springer-Verlag, Berlin (1976), 60-74.
4. G. Dahlquist, Positive functions and some applications to stability questions for numerical methods, preprint, MRC Symposium, Madison, Wisc., April 1978.
5. J. Douglas, On the numerical integration of $u_{xx} + u_{yy} = u_t$ by implicit methods, J. Soc. Indust. Appl. Math. 3 (1955), 42-65.
6. J. Douglas and J. E. Gunn, A general formulation of alternating direction methods, Numer. Math. 6 (1964), 428-453.
7. A. C. Hindmarsh, GEARB: Solution of ordinary differential equations having banded Jacobian, Lawrence Livermore Laboratory Report UCID-30059, Rev. 1, 1975.
8. W. Liniger and R. A. Willoughby, Efficient integration methods for stiff systems of ordinary differential equations, SIAM J. Numer. Anal. 7 (1970), 47-66.
9. H. Lomax, Stable implicit and explicit numerical methods for integrating quasi-linear differential equations with parasitic-stiff and parasitic-saddle eigenvalues, NASA TN D-4703, 1968.
10. D. W. Peaceman and H. H. Rachford, The numerical solution of parabolic and elliptic differential equations, J. Soc. Indust. Appl. Math. 3 (1955), 28-41.
11. T. H. Pulliam and J. L. Steger, On implicit finite-difference simulations of three dimensional flow, AIAA Paper 78-10, 1978.
12. J. L. Steger, Implicit finite difference simulation of flow about arbitrary two-dimensional geometries, AIAA J. 16 (1978), 679-686.
13. R. F. Warming and R. M. Beam, An extension of A-stability to alternating direction implicit methods, submitted to BIT. Also NASA TM 78537, 1978.

TOLERANCE PROPORTIONALITY IN ODE-CODES

Hans J. Stetter
Tech. Univ. Vienna

1. Meaning of Proportionality

Most routines for the numerical solution of

$$(1.1) \quad y'(t) = f(t, y(t)), \quad y(0) = y_0, \\ t \in [0, T], \quad y(t) \in \mathbb{R}^s,$$

request the specification of an accuracy level through a tolerance parameter $\delta \in \mathbb{R}$. The choice of δ determines the computation for a given problem (1.1) so that all computational quantities are dependent upon δ .

Originally, the numerical solution of (1.1) by a given code is only defined on a "grid" $G(\delta) = \{t_n; n=0(1)N\} \subset [0, T]$; by a suitable interpolation (see section 2) we may generate an approximation $\eta(t, \delta)$ defined for all $t \in [0, T]$, with global error $\epsilon(t, \delta) := \eta(t, \delta) - y(t)$. Tolerance proportionality means that

$$(1.2) \quad \epsilon(t, \delta) \approx \delta \cdot v(t),$$

with a function $v: [0, T] \rightarrow \mathbb{R}^s$ independent of δ , holds for an interval of δ -values including those commonly used. (1.2) permits an assessment of ϵ from the results of two computations with different tolerances δ .

A vaguer form of proportionality occurs in the Toronto Assessment Programs ([1], [2]). Here the efficiency of a code in finding an approximate solution of a given accuracy for a given problem (1.1) at a given point t is measured on the basis of

$$(1.3) \quad \|\epsilon(t, \delta)\| \approx \delta^{\bar{e}} \cdot \bar{v}(t),$$

where \bar{e} and $\bar{v}(t)$ are determined by a logarithmic least squares fit.

In the following, "proportionality" will refer to the stricter relation (1.2) unless specified. We will regard various effects in current ODE-codes for their impact on the achievement of proportionality.

2. Local equivalent of proportionality

The numerical solution of (1.1) by a one-pass forward step method cannot be controlled with a global strategy. But (1.2), with v unspecified, is not a global requirement, at least not for small δ : Within first order perturbation theory it is equivalent to (cf., e.g., [3]) the local requirement that $\eta(t, \delta)$ satisfies

$$(2.1) \quad \eta'(t, \delta) \approx f(t, \eta(t, \delta)) + \delta \cdot \gamma(t),$$

with $\gamma: [0, T] \rightarrow \mathbb{R}^s$ independent of δ . (2.1) can be controlled on a step-by-step basis.

For specified values η_{n-1} and $\eta_n \in \mathbb{R}^s$, there is a unique vector $u_n \in \mathbb{R}^s$ such that the solution of

$$z'(t) = f(t, z(t)) + u_n, \quad z(t_{n-1}) = \eta_{n-1},$$

satisfies $z(t_n) = \eta_n$. We assume that our continuous approximation $\eta(t, \delta)$ has been determined in this fashion; u_n may be called its "backward error" in (t_{n-1}, t_n) , cf. [4]. (2.1) requests

$$(2.2) \quad u_n(\delta) \approx \delta \cdot \gamma(t) \text{ in } [t_{n-1}, t_n].$$

It is well-known (see [3] or [4]) that u_n and the "local error per unit step" L_n are related by

$$(2.3) \quad L_n = u_n(1 + O(h_n));$$

thus proportionality requires L_n to be proportional to δ .

For a code "of order p ", with $L_n = O(h_n^p)$, this implies $h_n = O(\delta^{1/p})$. Thus, even if we manage to control the computation properly w.r.t. leading powers of h , we must expect $(1 + O(\delta^{1/p}))$ -factors in the proportionality relation (1.2) if we replace the \approx -sign by equality. Therefore we need not bother about $(1 + O(\delta))$ -factors which arise from linearization w.r.t. δ in (2.1) or from the evaluation of quantities along $y(t)$ in place of $\eta(t, \delta)$.

We assume that the actual control of the computation is effected through a vector $\hat{L}_n \in \mathbb{R}^s$ which is computed in each step: On the basis of

\hat{L}_n and δ , an acceptance procedure determines whether to accept the current step or not, and a step-size (and order) control procedure calculates the stepsize (and order) to be used in the next (or repeated) step.

3. Local Error Estimator Effects

In a one-step method of order p , we may compute \hat{L}_n from any procedure $\hat{L}(t_{n-1}, \eta_{n-1}, h_n; f)$ which satisfies

$$(3.1) \quad \hat{L}(t, y(t), h; f) = h^p \hat{\phi}(t+h)(1+O(h))$$

if we only achieve

$$(3.2) \quad \hat{L}_n = \delta \cdot \hat{\gamma}(t_n)(1+O(h_n))$$

through our control. For a backward error u_n of order p , (3.1)/(3.2) implies

$$\begin{aligned} (3.3) \quad u_n &= h_n^p \varphi(t_n) (1+O(h_n)) \\ &= h_n^p \hat{\phi}(t_n) \operatorname{diag} \left(\frac{\varphi(t_n)}{\hat{\phi}(t_n)} \right) (1+O(h_n)) \\ &= \delta \hat{\gamma}(t_n) \operatorname{diag} \left(\frac{\varphi(t_n)}{\hat{\phi}(t_n)} \right) (1+O(h_n)) \\ &= \delta \gamma(t_n)(1+O(h_n)), \end{aligned}$$

which is our requirement (2.2). The difficulty which may arise if a component of $\hat{\phi}$ vanishes will be considered in section 6. Note that the error estimate (3.1) need not be asymptotically correct, i.e. $\hat{\phi}$ may differ from the principal error function φ .

In a multistep PC-method, the above analysis remains valid only if at each step change new back values are computed by a sufficiently accurate interpolation procedure (of order $p+1$). If the previous grid remains unchanged, the expression for the backward error becomes

$$(3.4) \quad u_n = [P_1(h_n)\varphi_1(t_n) + P_2(h_n)\varphi_2(t_n)] (1+O(h))$$

where the polynomials $P_m(h)$ of degree p have h^2 -terms as their lowest powers so that $u_n = O(h^2)$ if the previous steps remain fixed; cf. e.g. [5]. Similarly, the commonly used local error estimators behave like $\hat{P}(h_n)\hat{\phi}(t_n)$, with $\hat{P}(h_n) = O(h_n^2)$ but $\hat{P} \neq P_m$, $m = 1, 2$. Only if we use an asymptotically correct local error estimator, which is far more expensive, we will have a parallel behavior of u_n and \hat{L}_n .

Thus, in a customary variable step multistep code (like STEP) even if we would manage to keep \hat{L}_n strictly proportional to δ this would not be fully true for u_n . As long as no violent step changes occur, the effects are not dramatic; but they add an other element of uncertainty to the use of "fictitious" (viz. lower order) local error estimates.

4. Acceptor Effects

We assume that the acceptance procedure decides on the basis

$$(4.1) \quad \begin{aligned} \|\hat{L}_n\| &\leq \delta && \text{accept} \\ \|\hat{L}_n\| &> \delta && \text{reject} \end{aligned}$$

where $\|\cdot\|$ may include a weighting, e.g. by the current solution values. (The type of weighting and the choice of \mathbb{R}^S -norm are supposed to be independent of δ .)

If we have a local error estimation of type (3.1), we have

$$(4.2) \quad \hat{L}_n = \frac{\|\hat{L}_n\|}{\|\hat{\phi}(t_n)\|} \hat{\phi}(t_n)(1+O(h_n));$$

thus the fact that we consider a norm only in (4.1) does not hurt, and the weighting may as well be assumed to use true solution values (cf. section 2) and thus to be completely independent of δ .

If we could deduce $\|\hat{L}_n\| \approx \delta$ from (4.1), (4.2) would immediately yield the relation (3.2) which was assumed in section 3. Actually, the fact that the usual acceptor does not contain a lower bound on $\|\hat{L}_n\|$ may seriously impair proportionality:

Consider a high order code which is conservative in increasing the stepsize so that values of $\|\hat{L}_n\|$ around 0.1δ do not lead to a change in stepsize for a long interval of integration. In this case a change of δ by a factor of 100 may not change the computation at all and the global error remains insensitive to changes of δ over a wide range of δ -values. When δ is further increased, the global error may suddenly jump to a much larger value.

Thus the inequality in (4.1) will permit an undermining of strict proportionality while it may not seriously affect the vague proportionality mentioned in section 1.

5. Stepsize Control Effects

Since the acceptor will not prevent $\|\hat{L}_n\|$ from assuming values far smaller than δ , the stepsize control must be designed such that $\|\hat{L}_n\|$ is kept close to δ . Actually, in view of (4.1), (4.2), and (3.2), a proper strategy is to aim at

$$(5.1) \quad \|\hat{L}_n\| \approx r\delta, \quad r < 1 \text{ fixed.}$$

For one-step methods it can be proved that the following procedure implies (5.1) under the assumption of (3.1), at least for smooth $\|\hat{\phi}(t)\|$:

$$(5.2) \quad h_{\text{new}} = \left(r \frac{\delta}{\|\hat{L}_n\|} \right)^{\frac{1}{p}} h_{\text{old}},$$

this remains true even if rejections occur due to a growth in $\|\hat{\phi}\|$.

Actually, (5.2) is the usual strategy for one-step codes where the stepsize may be changed virtually without cost; special provisions are added for very small and very large values of $\delta/\|\hat{L}_n\|$. Thus, at least at this fundamental point, strict proportionality is supportable by a reasonable code.

For multistep methods which use interpolation at a stepsize change, the same strategy (5.2) would be indicated; but due to the cost involved one will tend to introduce a threshold for changes of h . Such thresholds may seriously disturb the achievement of (5.1), particularly for high order methods.

For multistep methods which adapt their coefficients to the local grid structure, we have the difficulty exposed in section 3 (cf. (3.4)). Here it is unknown what a strategy maintaining (5.1) should look like, even if we could neglect the cost of changing h in each step. The usual strategies applied in such codes attempt to keep the stepsize constant over longer intervals and use (5.2) if a change has become necessary. It seems that this tends to maintain "vague proportionality" in the sense of section 1. Certainly, strict proportionality cannot generally be expected under these circumstances.

6. Problem Effects

So far, we have assumed that the problem (1.1) is sufficiently smooth to let the order of the method become effective, see (3.3) and (3.1). Let us now consider what happens if the computation crosses a finite jump discontinuity in f .

Assume that the local error estimate is based on a lower order local solution so that

$$(6.1) \quad \hat{L}_n = h_n \hat{1}_n, \quad \text{with} \quad \hat{1}_n = O(h_n^{p-1}).$$

In a wide class of codes, $\hat{1}_n$ is essentially a linear combination of evaluations of f ; if f has a jump along the solution trajectory, $\hat{1}_n$ consists of a "smooth part" which would have arisen with a smooth continuation of f and the contribution from the jump in f :

$$(6.2) \quad \hat{L}_n = h_n (\tilde{1}_n + \alpha \Delta f),$$

α is some factor which may depend on the position of the jump relative to the step.

If h_n has to be reduced because of $\|\hat{L}_n\| > \delta$, $\tilde{1}_n$ decreases like h_n^{p-1} and acceptance occurs essentially for

$$(6.3) \quad h_n \propto \|\Delta f\| \leq \delta$$

Similarly, the contribution of this step to the global error consists of two parts: The effect of the "smooth" backward error (normally very small due to the smallness of h_n) and an immediate increment in $e(t)$ due to a wrong consideration of Δf . This second error is of the form $h_n \beta \Delta f$, its effect on the global error further on is thus proportional to $h_n \|\Delta f\|$. Due to the irregular dependence of α

and β on the relative location of the singularity w.r.t. the current computation, the proportionality to δ which would otherwise follow from (6.3) (interpreted as $h_n \|\Delta f\| \approx \delta$) is weakened. But it is surprising that such a strong trace of proportionality can be shown to remain even after a jump in f .

Another problem dependent difficulty was pointed out in section 3: components of $\hat{\Phi}$ may pass through zero. Due to the restriction to a norm in (4.1) and (5.2) this becomes fully effective only for scalar problems (1.1); but some damage may also appear when a strongly dominant component of the local error estimate changes sign in a system (1.1).

Consider the scalar case, with $\hat{\Phi}(t) \approx a(t-\bar{t})$ in the vicinity of \bar{t} . Then (5.2) and (3.1) imply

$$h_n = \left| \frac{t_{n-1} - \bar{t}}{t_n - \bar{t}} \right|^{\frac{1}{p}} h_{n-1}$$

so that the stepsize will grow locally until $t_n > \bar{t}$; but the details of that growth will depend on n the relative location of \bar{t} in the grid and thus on δ in an irregular manner. Thus there will be large u_n in the vicinity of \bar{t} (cf. (3.3) and note that $\varphi(t) \neq 0$), but their size will generally not be proportional to δ . This may introduce a considerable perturbation of strict proportionality.

7. Starting Effects

The choice-of-starting-step dilemma is well-known: On the one hand, it is unwise - and contrary to the intentions of numerical software design - to delegate this choice to the user; on the other hand, the values of y_0 and $f(t_0, y_0)$ are simply not sufficient to let the code choose a reasonable first step, cf. e.g. [6].

From the point of view of achieving proportionality, it is extremely undesirable to have an initial interval in which the stepsize (and hence the local error) is considerably smaller than the specified tolerances would require; cf. section 4. The following suggestion seems to be practical:

Design a procedure which computes an h_1 from y_0 , $f(t_0, y_0)$, and δ which would be the proper stepsize if some heuristic assumptions about the problem were true. Execute the first step and test for acceptance. Repeat the step, with a new stepsize computed from (5.2), not only if $\|\hat{L}_1\| > \delta$ but also if $\|\hat{L}_1\| < c_0 \delta$, with a reasonable value $c_0 < 1$. Details may be found in [6].

Such a procedure will prevent ill-effects from unduly small starting stepsizes. The same procedure should also be used for restarting after a singular condition, like a jump in f , has been recognized by the code:

Assume that h has to be reduced to 10^{-4} in order that a discontinuity in f may be passed but that the smoothness of f after the jump permits steps of order 10^{-1} under the specified tolerance. With or-

binary stepsize control strategies, the computation will take many steps to return to the appropriate stepsize. Therefore a restart, even if it involves a repetition of the first step will be much more economic besides enhancing proportionality.

3. Output Effects

So far we have assumed that steps are chosen exclusively to comply with local error requirements (except in "emergency situations"). In many codes, however, these requirements are overridden by the requirement to reach output points precisely. If there is a majority of steps determined in this fashion, proportionality must obviously become completely obsolete.

But even if we consider output at the end T of an integration interval only, the situation is not trivial. A common and natural strategy is the following: Determine h_{n+1} by the local error vs. tolerance requirements; use the stepsize

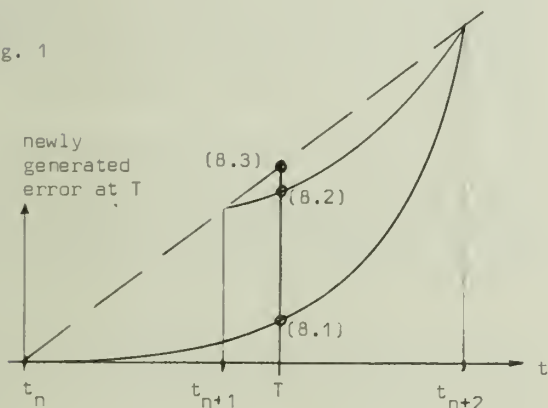
$$(8.1) \quad \bar{h}_{n+1} := \begin{cases} T - t_n & \text{if } T \in (t_n, t_n + h_{n+1}], \\ (T - t_n)/2 & \text{if } T \in (t_n + h_{n+1}, t_n + 2h_{n+1}), \\ h_{n+1} & \text{if } T \geq t_n + 2h_{n+1}. \end{cases}$$

Thus normally the last two steps before T are modified.

A simple analysis shows that - from the point of view of proportionality - it is better not to look ahead beyond $t_n + h_{n+1}$:

$$(8.2) \quad \bar{h}_{n+1} := \begin{cases} T - t_n & \text{if } T \in (t_n, t_n + h_{n+1}), \\ h_{n+1} & \text{if } T \geq t_n + h_{n+1}. \end{cases}$$

fig. 1



This is best explained by fig.1. Assume $h_{n+1} < T - t_n < h_{n+1} + h_{n+2}$, where h_{n+1} and h_{n+2} are defined by local error requirements. According to (2.1)/(2.2), the newly generated error in $[t_n, T]$ should be

$$(8.3) \quad \delta(T - t_n) \gamma(T) (1 + O(h_{n+1}))$$

which is indicated by the broken line in fig.1, without the $(1 + O(h_{n+1}))$ -factor. The errors generated under (8.1) and (8.2) are shown. It may be argued that the differences are $O(h_{n+1} \delta)$. But this is the leading power on the level $n+1$ of the newly generated error per step.

Also in codes where the output values are obtained by interpolation, care has to be taken if strict proportionality is not to be abandoned in the output process. Fig.2 shows the source of the difficulty. Some related analysis may be found in [4].

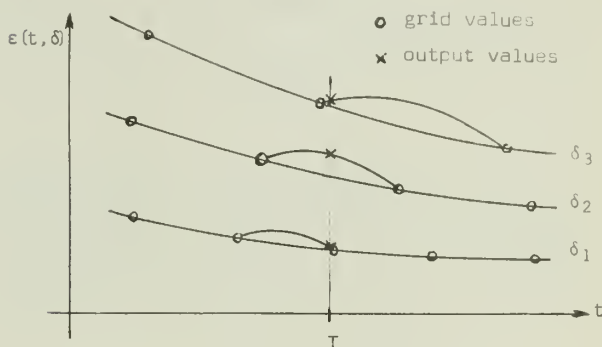


fig. 2

9. Mild Stiffness Effects

We do not wish to consider here the chances for proportionality in the application of a "stiff code" to stiff systems (1.1). There are some particular aspects of this situation which require a special analysis: A constant error-per-unit-step control seems not to be advisable (see [7]) and the damping of transient errors may be stronger for large h in L-stable methods.

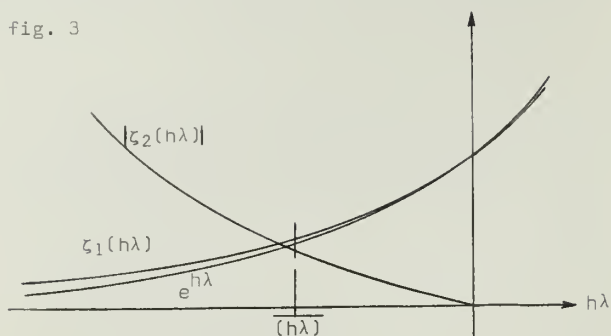
Here we consider a typical "non-stiff", i.e. explicit code and a mild stiffness in (1.1). At a crude tolerance which permits relatively large steps, our basic assumption (3.3) on the backward error in a p -th order method may no longer hold. The typical situation in a multistep method is shown in fig.3. (only real negative eigenvalues are considered).

If the local error requirement applied to the principal root ζ_1 permits $h\lambda < (\overline{h\lambda})$ for the dominant or for a subdominant component, a "parasitic" contribution to the error will grow until its effect is large enough to affect the stepsize control. Although the reaction of the control will not be appropriate (since it is geared on a h^p -behavior), $h\lambda$ will eventually move to the safe interval $h\lambda > (\overline{h\lambda})$. Here the local error is too small so that h is increased again. The same effects occur in one-step methods if we consider the one (principal) root for a dominant and a (mildly stiff) subdominant component of the solution.

We may say that in such a situation the computation is controlled by stability rather than by accuracy. Since the stability requirement has nothing to do with the specified tolerance δ , a computation which is predominantly controlled by sta-

bility cannot be expected to yield a global error proportional to δ .

fig. 3



We may even define a given problem (1.1) to be stiff for a given non-stiff code at a specified tolerance, if the computation is no longer controlled by local error requirements throughout. In a high-order Adams PECE-code stiffness in this sense should be very common as a quantitative analysis shows:

Define $\overline{(h\lambda)}$ by

$$(9.1) \quad |\zeta_{\text{paras.}}(h\lambda)| \geq .9 e^{h\lambda} \text{ for } h\lambda < \overline{(h\lambda)}$$

and let us characterize the location of $\overline{(h\lambda)}$ by the corresponding number of "locally correct digits"

$$-10 \log \frac{\zeta_{\text{princ.}}(\overline{(h\lambda)}) - e^{\overline{(h\lambda)}}}{e^{\overline{(h\lambda)}}}.$$

Then a situation is stiff for a k-step Adams PECE-code ($p=k+1$) if the tolerance requests fewer than

k	4	5	6	7	8	9	10
locally correct digits	1.5	2.5	3.75	5	6.5	8	9.5

10. Change of order (method)

So far we have made the silent assumption that the tolerance parameter δ influences the computation through the selection of the stepsize only. However, some of the most efficient codes are based on varying order or varying basic integration procedures.

This is well-known for Adams multistep codes; the recent construction of families of embedded Runge-Kutta procedures ([9]) suggests a similar approach for one-step methods. This approach is further supported by results which demonstrate that lower order methods are more efficient at crude tolerances ([4]).

In a "variable procedure" code we can, of course, not assume that the same procedure is used in a certain section of the integration interval for different δ . This affects our considerations at their roots:

In section 3, we were able to permit the fictitious local error estimate (3.1), with $\hat{\phi} \neq 0$, because we could assume $\hat{\phi}(t)$ to be essentially independent of δ ; thus a function γ was generated in (2.2) which was also essentially independent of δ . If there may be different functions $\hat{\phi}$ for different δ , the argument in section 3 breaks down.

Furthermore, even if we have an asymptotically correct error estimator and achieve $\|u_n\| \approx \delta$ for all n , this does not imply (2.2) in the variable procedure case. Different vectors $\gamma(t)$ of identical norm may lead to quite different solutions of (2.1).

Thus there is not much to expect with regard to proportionality from a variable procedure code, no matter whether it uses local extrapolation or not. On the other hand, this does not mean that proportionality, at least of the vague type, cannot occur with such codes: If there is sufficient regularity in the relation between the various high order derivatives, the resulting function value $\gamma(t)$ in (2.2) and (2.1) may not depend strongly on the particular procedure which is used in the neighborhood of t .

Actually, it has been experimentally established that, e.g., the variable order, variable step multistep code STEP which also uses local extrapolation exhibits a considerable degree of vague proportionality for a wide variety of problems.

11. Conclusions

The preceding considerations should have made it clear that the output of a robust and efficient code for initial value problems (1.1) cannot be expected to exhibit more than the vague proportionality between the global error and the specified tolerance δ which has been described in section 1. An exponent $\tilde{e} \approx 1$ in (1.3) should appear only if the least squares fit is based on a wide range of tolerances.

If δ is varied only over a narrow range, with many samples taken, say $\delta = 10^{-3(.1)^5}$, and if the values of the individual error components are regarded in place of the norm of the error vector, we should not be surprised if few traces of proportionality are observable before smoothing.

This means that we cannot reliably judge the size of the global error from running the problem twice with different tolerances. But this was the main advantage that we hoped to obtain. If we cannot have it we may as well forget about proportionality, at least from a utilitarian point of view.

Even if our codes were more perfectly proportional, the ordinary user would certainly not bother to run them twice at different tolerance levels if they would give him an approximate value of the global error directly. I do not believe that he would be very fastidious w.r.t. the accuracy of this

approximate value; the order of magnitude might be sufficient in many cases. He might even accept an occasional blunder without much fuss. (Precisely this behavior is shown by the users of quadrature codes where the reliability of the error assessment is no better.)

If we admit that an assessment of the global error should be incorporated into ODE-codes, the question is whether it can be done more efficiently by making the codes more proportional or by a direct approach. From my experiences, I would favor the second choice; see [10].

Nevertheless, the analysis of the proportionality properties of ODE-codes yields important insights into the mechanism of these codes.

References

- [1] W.H. Enright: Using a testing package for the automatic assessment of numerical methods for ODEs, in: Performance Evaluation of Numerical Software, Proceedings, North Holland, to appear
- [2] W.H. Enright, T.E. Hull: A collection of programs for assessing initial value methods, Dept. Comp. Sci. Tech.Rep., Univ. of Toronto, 1979.
- [3] H.J. Stetter: Considerations concerning a theory for ODE-solvers, in: Numerical Treatment of Differential Equations, Springer Lecture Notes Vol.631 (1978) 188 - 200.
- [4] K.R. Jackson, W.H. Enright, T.E. Hull: A theoretical criterion for comparing Runge-Kutta formulas, SINUM 15 (1978) 618 - 641.
- [5] H.J. Stetter: Interpolation and error estimation in Adams PC-codes, SINUM 16 (1979).
- [6] L.F. Shampine, H.J. Stetter: Starting an ODE-solver, to appear.
- [7] B. Lindberg: Characterization of optimal step-size for methods for stiff differential equations, SINUM 14 (1977) 859 - 887.
- [8] H.J. Stetter: On the maximal order in PC-codes, Computing 20 (1978) 273 - 278.
- [9] D.G. Bettis: Efficient embedded Runge-Kutta-methods, in: Numerical Treatment of Differential Equations, Springer Lecture Notes Vol.631 (1978) 9 - 18.
- [10] H.J. Stetter: Global error estimation in ODE-solvers, in: Numerical Analysis, Springer Lecture Notes Vol.630 (1978) 179 - 189.

Numerical solution of large systems of ODE's with sparse Jacobians

by

Per Grove Thomsen

Department of Numerical Analysis

Technical University of Denmark.

Introduction.

In many technical applications, model problems are defined by systems of ordinary differential equations

$$y' = f(t, y) \quad , \quad t \in [0, T] \quad , \quad y \in \mathbb{R}^n \quad (1)$$

where n is a large number, and where the Jacobian matrix

$$J = \left[\frac{\partial f_i}{\partial y_j} \right] \quad i, j = 1, 2, \dots, n \quad (2)$$

is a sparse matrix. In this paper we shall discuss the problem of selecting a solution method and show how sparse matrix techniques can be used to solve the systems of linear equations that arise, and how it may be implemented efficiently.

Solution methods.

The systems (1) that concern engineers will usually be characterised by having a wide spectrum of eigenvalues for the Jacobian J , and these will often vary along the solution. This means that only methods for stiff systems can be used in general purpose software. On the other hand the engineer will often have very modest requirements of accuracy. In such cases methods of constant order will be reasonably efficient, for the present discussion we have selected order two.

Often it will be advantageous for the stepsize strategy to take large steps in spite of the stiff character of the system, therefore it is important that the region of absolute stability of the method contains points at infinity in the left half plane, and that damping of errors in the stiff components will take place. In some applications one would also be interested in damping of highly oscillatory

components.

These objectives can be achieved if, in the case of one-step methods we have $L(\alpha)$ -stability, while in the case of linear multistep methods we could use the following $L_k(\alpha)$ -stability definition: (see ref.[1]).

Definition: A linear k -step method

$$\sum_{j=0}^k \alpha_j y_{n+j} = h \sum_{j=0}^k \beta_j f_{n+j}$$

is $L_k(\alpha)$ -stable if (i) it is $A(\alpha)$ -stable, and if applied to the test equation $y' = \lambda y$, $\text{Re}(\lambda) < 0$, it yields

$$|y_{n+k}| \leq R(h\lambda) \max_i |y_{n+i}| \quad , \quad 0 \leq i \leq k-1 \quad ,$$

such that $|R(h\lambda)| \rightarrow 0$ when $|\lambda| \rightarrow \infty$,
 $|\arg(-\lambda)| < \alpha$.

Of the methods that satisfies the above requirements we have considered two types, the Backward Differentiation methods and Semi-Implicit Runge-Kutta methods, for order two we will use the following formulae.

$$y_{n+1} = \frac{4}{3} y_n - \frac{1}{3} y_{n-1} + h \frac{2}{3} f_{n+1} \quad (3)$$

for error estimation, an approximation to y''' may be found by using backward differencing. The SIRK-methods are of Nørsett-type (see ref.[2]).

$$k_1 = f(t_n + \gamma h, y_n + \gamma h k_1) \quad (4)$$

$$k_2 = f(t_n + \delta_2 h, y_n + \beta_2 k_1 + \gamma h k_2)$$

$$k_3 = f(t_n + \delta_3 h, y_n + \beta_3 k_1 + \beta_3 k_2 + \gamma h k_3)$$

$$y_{n+1} = y_n + h(w_1 \cdot k_1 + w_2 \cdot k_2 + w_3 \cdot k_3) \quad (5)$$

$$LE = h(c_1 \cdot k_1 + c_2 \cdot k_2 + c_3 \cdot k_3) \quad (6)$$

Implementation.

In a variable stepsize implementation the step-sizes are in general controlled by the estimated local errors. In the stepsize selection strategy the error constants will influence the stepsizes chosen in such a way that the smaller the error constant is, the larger the stepsize will be chosen. In our comparison the SIRK will have a smaller error constant than the BDF method. If the same error tolerance is prescribed we may expect the SIRK to do better on average than the BDF because the number of steps will be smaller. On the other hand, Nørsett has shown (ref.[2]), that it is possible to select the parameters in (4,5,6) in such a way that the k_3 found in step n can be used as k_1 in step $n+1$, so that only two k_i 's have to be found in each step, except when starting and when changing stepsize.

Each of the formulae (3) and (4) is a nonlinear system of equations that has to be solved in each step, this solution is in general solved by means of quasi-newton iterations, that take the following forms,

in the BDF case:

$$(I - \beta h J) d_{n+1}^{[s]} = (y_{n+1}^{[s]} - \frac{4}{3} y_n + \frac{1}{3} y_{n-1} - \frac{2}{3} h f_{n+1}^{[s]}) \quad (7)$$

$$y_{n+1}^{[s+1]} = y_{n+1}^{[s]} - d_{n+1}^{[s]} \quad (8)$$

in the SIRK case:

$$(I - \gamma h J) e_i^{[s]} = k_i^{[s]} - f(y_n + \gamma h k_i^{[s]}) \quad (9)$$

$$k_i^{[s+1]} = k_i^{[s]} - e_i^{[s]} \quad (10)$$

In each case J is an approximation to the Jacobian, this is only updated in case the convergence is too slow. It is convenient to consider the process written in the form

$$A_r \delta_{n+1}^{[s]} = b_{n+1}^{[s]} \quad (11)$$

where $A_r = (I - \gamma h J)$ and $b_{n+1}^{[s]}$ is one of the right hand sides (7) or (9) both formulae have this general form. If we consider the time history of the solution we will find a sequence of matrices A_r , $r = 0, 1, \dots$ that can be displayed in the form

$$\begin{array}{ccccccc} A_0 & \cdot & \cdot & \cdot & A_1 & \cdot & \cdot & \cdot & A_2 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & A_r \cdot \\ t_0 & t_1 & t_2 & & & & & & & & & & & & & & & t \end{array}$$

The matrices A_r will be changed when either the stepsize is changed or when the convergence is slow and the Jacobian updated. If our system is large the factorizations of the matrices A_r will be very expensive and the main part of the total work involved is spent on that. If sparse techniques is used the work can be reduced but it is essential that as much information is carried along about the earlier factorizations.

Application of sparse techniques.

The subroutine SSLEST has been constructed so as to facilitate the solutions in such cases. By using ordered lists for storing only nonzero elements and by selecting pivots so that the number of fill-in's is small, each factorization is made very efficient. Further if as in the case above the sequence of matrices A_r all have the same structure, then the information in the pivotal strategy may be used in subsequent factorizations, thereby saving a great amount of work, in general this amount will be about 60% of the total factorization. This sparse matrix package has been used in connection with the SIRK method in a subroutine SPARKS that will be presented, and compared to the use of BDF methods.

Conclusions.

The use of sparse matrix techniques in connection with methods for stiff systems for ODE's with sparse Jacobians, can be very effective. Two types of methods have been compared, the BDF methods versus the SIRK methods, both are sufficiently stable to allow the use of large step-sizes. Efficient implementations with a sparse package is presented.

References.

- [1]: Z. Zlatev, P.G. Thomsen: An analysis of the solution of linear systems of ordinary differential equations. Inst. for Num. Anal., Technical University of Denmark, June 1977, NI-77-10.
- [2]: S.P. Nørsett: Semi Explicit Runge Kutta methods, Mathematics and Computation No. 6/74. Dept. of Mathematics, University of Trondheim, ISBN 82-7151-009-6.
- [3]: Z. Zlatev, V.A. Barker, P.G. Thomsen: SSLEST A Fortran subroutine for solving sparse systems of linear equations, Users guide. Inst. for Num. Anal., Technical University of Denmark, NI-78-01.

AUTOMATIC PARTITIONING OF STIFF SYSTEMS AND EXPLOITING THE RESULTING STRUCTURE

W.H. Enright
and
M.S. Kamel

Department of Computer Science
University of Toronto

Abstract

Most methods for solving stiff systems are based on implicit formulas and require the use of Newton-like iterations. The cost of the matrix operations in the iteration scheme of these methods can be quite high.

A new iteration scheme is developed which exploits the structure of the system and also allows fast updating of the iteration matrix after a stepsize or order change. The technique is particularly useful when the stiffness is due to only a few components of a large system, and is applicable to most methods for stiff systems. It is based on an automatic partitioning of the system into subsystems corresponding to stiff and non-stiff parts. This partitioning is then exploited in the solution of the system of equations that must be solved on each time step.

1. Introduction

In most methods for solving stiff systems, the solution of a system of equations on each time step by a modified Newton iteration usually consumes a considerable amount of the computational effort of the method.

One way to achieve efficiency is to take advantage of the structure of the problems and develop techniques to solve problems of the same structure. For example, problems with banded or sparse Jacobians can be solved efficiently by methods with iteration schemes based on banded or sparse (linear system solvers), rather than using general methods (see Hindmarsh [1976]).

If the system can be partitioned into stiff and non-stiff parts (transient and smooth components), more effective techniques may be developed to reduce the cost of the iteration scheme. The idea of partitioning the system into transient and smooth parts is an old idea and is related to the singular perturbation problem, where the system

$$\dot{y}(t) = f(y(t)), \quad y(0) = y_0 \quad (1.1)$$

is transformed into

$$\begin{pmatrix} D\dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} f_1(u, v) \\ f_2(u, v) \end{pmatrix} \quad (1.2)$$

This research was supported by the National Research Council of Canada

where

D is a matrix which may be singular
 u corresponds to transient components
 v corresponds to smooth components.

As early as 1952, Dahlquist used a method called Pseudo Stationary Approximation (PSA) in which the transient components are eliminated by putting $D=0$, thus reducing the system to a differential algebraic system. The integration can then be done by any conventional ODE solver. This method was revised by Dahlquist [1969] and analysed and tested by Oden [1971]. The new version is called SAPS (for smooth approximate particular solution).

Partitioning of linear systems of the form, $\dot{y} = Ay + b(t)$, where A is a matrix with constant elements and with the large eigenvalues well separated from the small eigenvalues, has been considered by Lee [1967] and by Branin [1967] in connection with circuit theory.

Lee used matrix filtering to divide the system into loosely coupled subsystems, one for the smooth components and one for the transients. The smooth and transient components are obtained by using the transformation

$$y_1 = LP y, \quad y_2 = HP y;$$

where LP and HP are matrices which act like filters. LP passes the eigenvectors of A that correspond to small eigenvalues and annihilates eigenvectors that correspond to large eigenvalues. HP has the opposite effect. The integration procedure he used is similar to the PSA method where the transient components are eliminated (by putting $\dot{y}_2 = 0$) and the smooth components

are integrated by a conventional method.

Branin suggests that the matrix A be decomposed into a sum of diagonal and off-diagonal parts. If the diagonal part has large eigenvalues and the off-diagonal part has small eigenvalues, the restriction on the stepsize for the formula he used (which is based on an analytical solution) is relaxed. He noted that, for most practical problems, the matrix A has to be transformed first so it can be decomposed in this way. Branin suggests using some cycles of the QR-algorithm to do the transformation. He also gives guidelines to handle nonlinear systems, where the Jacobian matrix plays the role of the matrix A , but is changing during the course of the integration. In this case the transformation has to be frequently updated.

Methods based on implicit formulas have been widely used for solving stiff systems. However, using these methods may not be efficient for large systems in which the stiffness is caused by only a few components of the system. The cost of dealing with the Jacobian of the full system can be quite high. Dahlquist has pointed out that more work should be done on designing software which can utilize the favorable situation which occurs when some subsystems are not stiff (Dahlquist [1974]).

Alfeld and Lambert [1977] report on a method where the correction is only done in the 'dominant' space (the space spanned by the transient components). They consider systems of the form, $\dot{y} = A(t)y + g(t)$, where the eigenvalues of $A(t)$ may be separated into two parts, one of which dominates the other. The integration consists of taking one forward step by a conventional multistep method and then making a correction in the subspace spanned by the eigenvectors corresponding to the dominant eigenvalues. The technique requires the explicit computation of the dominant eigensystem. Application to nonlinear systems is also discussed where the eigensystem of the Jacobian will be needed at each step.

Recent work on partitioning stiff system has also been reported by Hoffer [1976]. Hoffer's method is designed to solve large systems in which the transient components originate from only a few, easily identified equations. His integration formula is a combination of the modified midpoint rule applied to the subsystem of the smooth components, and the implicit trapezoidal rule applied to the transient components.

In the partitioning techniques described above the difficulty of solving linear systems of equations on each time step is reduced somewhat by applying an analytic expansion or implicit formula (with the modified Newton iteration) only to the stiff subsystem and applying a standard non-stiff formula to the remainder of the system. The accuracy and stability of such techniques is very sensitive to the correct identification of the corresponding subsystem as well as to the amount of coupling that is present between these subsystems. It should also be noted that, even for linear problems, the appropriate partitioning will in general be time and tolerance dependent. The technique we propose is based on applying the same implicit formula to all components of the system and exploiting partitioning only in the iterative solution of the resulting systems of equations. As a result only the cost of the iteration will be sensitive to the correct identification of the corresponding subsystem and the accuracy and stability of the integration will not be sensitive to the partitioning.

In a related study, Robertson [1976] has considered partitioning of the iteration matrix in the iteration scheme. By fixing the part of the iteration matrix which corresponds to the smooth components and only updating the part which corresponds to the transient components, the iteration scheme will result in using a modified-Newton technique only on transients and repeated substitution with correction terms for smooth components. By allowing periodic updating of the part corresponding to the smooth components further efficiency can be achieved. Robertson

does not discuss how one can obtain the appropriate partitioning.

Techniques that achieve efficiency in solving stiff systems without pre-information about the transient and the smooth components are more general and can easily be incorporated into existing packages for stiff systems. The fast update technique developed by Enright [1978] is of this kind. The iteration matrix is reduced to Hessenberg form and subsequent updates of the representation of the iteration matrix, after stepsize or order changes, require fewer operations than working with the inverse of the iteration matrix, or its LU factors.

The technique to be developed and analyzed in this study is related to Enright's approach. As the iteration matrix is reduced using a sequence of similarity transformations, a test is performed to identify the stiff components. The resulting partitioning of the system leads to an iteration scheme which adapts to the behaviour of the problem and can be interpreted as applying a modified Newton iteration to the transient components and repeated substitution to the smooth components. (The idea is actually similar in spirit to the identification and solution of rank deficient least square problems by the method described in Lawson and Hanson [1974, pp. 77]). Substantial saving of computational operations can be expected, especially if the stiffness is due to only a few components of a large system.

2. The Proposed Technique

The application of an implicit multistep formula to the system

$$\dot{y} = f(y), \quad y(t_0) = y_0, \quad (2.1)$$

can be written as

$$0 = G(y_{i+1}) = y_{i+1} - h\beta_0 f(y_{i+1}) - \sum_{j=1}^k \alpha_j y_{i+1-j} - h \sum_{j=1}^k \beta_j \dot{y}_{i+1-j}. \quad (2.2)$$

An iteration scheme for solving the nonlinear system $G(y_{i+1}) = 0$, can be viewed as:

Predict y_{i+1}^0 ,

set $\delta_{i+1}^\ell = y_{i+1}^\ell - y_{i+1}^{\ell-1}$ for $\ell = 1, 2, \dots$, and

determine δ_{i+1}^ℓ by solving the linear system

$$W\delta_{i+1}^\ell = G(y_{i+1}^{\ell-1}) \text{ for } \ell = 1, 2, \dots \quad (2.3)$$

Different iteration schemes will result from different choices of W . For example, if W is the identity matrix then the iteration corresponds to repeated substitution or, in our context, a predictor corrector iteration; while if $W = [I - h\beta_0 A]$ where $A = \partial f / \partial y$ evaluated at

y_j^0 ($j < i+1$), the iteration corresponds to a modified Newton scheme.

If (2.1) is stiff then there is an obvious trade-off between these two possible choices of W . The former choice is cheap ($O(n)$ operation

per iteration) but in order to guarantee convergence the stepsize must be severely restricted ($||h\beta_0 A|| < 1$). On the other hand, the latter choice is expensive ($O(n^2)$ operation per iteration)

and $O(n^3)$ operations when W has to be updated), but much larger stepsizes are possible. Basically the scheme we describe is a compromise between these two choices of W . Our idea is to approximate the matrix A , of the Newton iteration by a matrix \tilde{A} that will permit efficient iterations and updates, and at the same time, include enough information for the iteration to converge.

Before we discuss the technique we should review the technique suggested by Enright [1978] for implementing the modified Newton iteration. He observed that if the system is of dimension n and $W = [I - h\beta_0 A]$ is retained in its LU factored

form then $O(n^2)$ operations will be required whenever the scalar $h\beta_0$ changes. On the other hand

if W is retained in its LHL^{-1} factorization (H is an upper Hessenberg matrix), then fast updates are possible when $h\beta_0$ changes and only when A

needs to be updated will W require a full update.

In order to motivate how we choose \tilde{A} , let us rewrite the modified Newton iteration as:

$$(A - (1/h\beta_0)I)\delta_{i+1}^\ell = (1/h\beta_0)G(y_{i+1}^{\ell-1}) \quad (2.4)$$

Consider the factorization of A by the similarity transformation:

$$A = SRS^{-1} \quad (2.5)$$

where R is of the form

$$R = \begin{pmatrix} H & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

and, H is an $k \times k$ upper Hessenberg, R_{12} is an $k \times (n-k)$ matrix, R_{22} is an $(n-k) \times (n-k+1)$ matrix.

The matrix R is of the same structure as the reduced matrix, A_k , at the k -th step in the reduction of A to upper Hessenberg form. That is, if the reduction were done using Householder transformations then

$$A_k = U_{k-1}U_{k-2} \dots U_1AU_1 \dots U_{k-2}U_{k-1}$$

where U_i , $i = 1, \dots, k-1$ are elementary reflectors.

We modify this approach by introducing pivoting (row and column interchange on each step). R is then the reduced matrix at the k -th step:

$$R = U_{k-1}P_{k-1} \dots U_1P_1AP_1U_1 \dots P_{k-1}U_{k-1} \quad (2.6)$$

where P_i is a permutation matrix.

Let $P_iU_i = S_i$. Then the matrix S in (2.5) will be

$$S = S_1 \dots S_{k-1}$$

and $S^{-1} = S^T$.

One would prefer, in our application, that the $k \times k$ matrix, H , correspond to the transient components. The interchanges are designed to

accomplish this by making $||R_{22}||$ small.

If $||R_{22}||$ is small enough, then we can replace R by the matrix

$$\tilde{R} = \begin{pmatrix} H & R_{12} \\ 0 & 0 \end{pmatrix} \quad (2.7)$$

and A can then be replaced by

$$\tilde{A} = S\tilde{R}S^{-1}.$$

Note that $||A - \tilde{A}||_F < ||R_{22}||_F$.

Returning to the system (2.4) and substituting \tilde{A} for A we get

$$\begin{aligned} (\tilde{A} - (1/h\beta_0)I)\delta_{i+1}^\ell &= S(\tilde{R} - (1/h\beta_0)I)S^{-1}\delta_{i+1}^\ell = \\ (1/h\beta_0)G(y_{i+1}^{\ell-1}) \end{aligned} \quad (2.8)$$

A decomposition algorithm has been developed which applies the transformation as in (2.6). The interchange of rows and columns is designed so that the columns of R_{22} with large norms will be the pivot columns. The value of k is determined as the smallest integer such that $||R_{22}||_F$ is less than a specified threshold value, T . The cost of this decomposition is approximately $10/3 n^2 k - 7/3 nk^2 + 4/3 k^3$ adds and multiplies.

The matrix H is updated by adding $(-1/h\beta_0)I$ and the resulting matrix is factored into its LU decomposition at a cost of $k^2/2$ adds and multiplies. Then, to solve (2.8) we must apply the following four steps:

1. form the vector $c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = S^{-1}(1/h\beta_0)G(y_{i+1}^\ell)$
2. set $z_2 = -(h\beta_0)c_2$
3. solve $Ly_1 = c_1 - R_{12}z_2$
4. solve $Uz_1 = y_1$
5. set $\delta_{i+1}^\ell = S \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$.

The cost of solving one system is then approximately $5nk - 5/2k$ adds and multiplies.

In the remainder of this paper we will discuss the selection of the threshold value, T and the application of this technique to existing methods. We will also present some numerical results illustrating the performance of this technique.

References

- P. Alfeld and J.D. Lambert [1977]. Correction in the dominant space: A numerical technique for a certain class of stiff initial value problems, Math. Comp., vol. 11, No. 140, pp. 922-938.
- F.H. Branin [1967]. Computer methods of network analysis, Proc. IEEE 55, No. 11, pp. 1787-1801.
- G. Dahlquist [1969]. A numerical method for some ordinary differential equations with large Lipschitz constants, in Information Processing 68, ed. A. Morell, North-Holland Pub. Co., Amsterdam, pp. 183-186.
- G. Dahlquist [1974]. Problems related to the numerical treatment of stiff differential equations, Proc. International Computing Symposium 1973, eds. A. Gunther et al., North-Holland Pub. Co., Amsterdam, pp. 307-314.
- W.H. Enright [1978]. Improving the efficiency of matrix operations in the numerical solution of stiff ODE's, ACM TOMS, 4, No. 2 pp.127-136.
- A.C. Hindmarsh [1976]. The LLL family of ordinary differential equations solvers. Lawrence Livermore Laboratory, Report UCRL-78129.
- E. Hoffer [1976]. Partially implicit method for large stiff system of ODE's with only few equations introducing small time constants, SIAM J. Num. Anal. 13, No. 5, pp. 645-664.
- C.L. Lawson and R.J. Hanson [1974]. Solving least squares problems, Prentice Hall, Englewood Cliffs, NJ
- H.B. Lee [1967]. Matrix filtering as an aid to numerical integration, Proc. IEEE, vol. 55, No. 11, pp. 1826-1831.
- L. Oden [1971]. An experimental and theoretical analysis of the SAPS method for stiff ordinary differential equations, Dept. of Information Processing, Rep. NA 71.28, Royal Inst. of Tech., Stockholm.
- H.H. Robertson [1976]. Numerical integration of systems of stiff ordinary differential equations with special structure, J. Inst. Math. Appl. 18, No. 2, pp. 249-253.

IMPLEMENTATION OF IMPLICIT FORMULAS
FOR THE SOLUTION OF ODES

L. F. Shampine
Applied Mathematics Research Department
Sandia Laboratories
Albuquerque, New Mexico

The material presented in the talk appears in written form as the report SAND79-0694 of the same title. Copies may be obtained from the author on request.

Implicit formulas are quite popular for the solution of ODEs. They seem to be necessary for the solution of stiff problems. Every code based on an implicit formula must deal with certain tasks studied in this paper: (i) A choice of basic variable has to be made. The literature is extremely confusing as to what the possibilities are and the consequences of the choice. These matters are clarified. (ii) A test for convergence of the method for solving the implicit equations must be made. Ways of improving the reliability of this test are studied. (iii) Deciding when to form a new approximate Jacobian and/or iteration matrix is a crucial issue for the efficiency of a code. New insight which suggests rather specific actions will be developed.

The report closes with an interesting numerical example. It was created to illustrate in a simple way a difficulty observed in practice. It also shows that in very reasonable circumstances, (i) the implicit formula can have more than one solution, (ii) the predictor can be closer to the "wrong" solution, and (iii) any convergence test based solely on the difference of successive iterates can accept an approximate solution from a divergent iteration.

TAYLOR SERIES METHODS WITH VARIABLE STEPSIZE RATIO AND VARIABLE ORDER

George Corliss
Dept. of Math. and Stat.
Marquette University
Milwaukee, WI 53233

Y. F. Chang
Dept. of Computer Science
University of Nebraska
Lincoln, NE 68588

Partial support for this work was provided by the National
Science Foundation under Grant No. MCS78-03022.

Taylor series methods compute a solution to an initial value problem in ordinary differential equations by expanding the solution in a long series. This solution is extended across the desired interval by analytic continuation. We present techniques for choosing the largest ratio of the integration stepsize to the series radius of convergence (R_c) which may be taken subject to error control constraints. This optimal ratio depends on the length of the series, the orders and locations of the primary singularities, the desired error tolerance (ϵ) and the type of error measure the user wishes to control. Then a series length can be chosen which minimizes the computational cost of solving the problem.

Most state-of-the-art codes for solving ODE's vary both the stepsize and the order of the formula used in order to handle efficiently a variety of problems and a range of error tolerances. Several authors have written translator programs which accept the statement of the system of ODE's to be solved and produce object code which can run later to solve the system using series methods. The lengths of the series used varied from 11 [1] up to 32 [5], but none of these codes attempted to determine the most efficient series length at run time. We have successfully solved examples using series as long as 1000 terms. Chang [2] gave an example for which 20 terms was found empirically to be a good length. Each author varied the stepsize to keep it substantially less than some estimate for R_c to control the error. By using a relatively long (20 to 30 terms) series, Chang was able to give an accurate estimate of R_c . Then a step of approximately $R_c \cdot \epsilon^{1/(N-1)}$ was used.

The choices of an optimal stepsize and series length require the accurate estimation of R_c from tests involving several consecutive terms of the series outlined in [3].

Singularities in the solutions to real ODE's can occur with any order, but they must occur either on the real axis or in conjugate pairs. Without loss of generality, the solution $y(x)$ may be expanded at $x = 0$ with series
$$\sum_{j=0}^{\infty} y^{(j)}(0) \frac{h^j}{j!} = \sum_{j=0}^{\infty} y_{j+1}. \quad \text{We assume for the moment that there is only one singularity on the circle of convergence. Solutions with more than}$$

one primary singularity are considered later in this paper. If the distance to the closest secondary singularity is R_2 , then the contribution from the secondary singularities to the series is $O(|R_c/R_2|^N)$ as $N \rightarrow \infty$. Hence the effects are negligible if a series is long enough. Consequently we consider the model problem $v(z; a, s) = (a - z)^{-s}$, for $s \neq 0, -1, \dots$.

Let $v(z; a, s) = \sum_{j=1}^N V_j$ be expanded using a step which satisfies $0 < r = h/a < 1$. Let N be large enough to satisfy $|\frac{N-1+s}{N} \cdot \frac{h}{a}| < 1$,
$$T_N = v(z) - \sum_{j=1}^N V_j, \quad d = \frac{N-1+s}{N}, \quad A = rV_N/(1-r),$$

and $B = rdV_N/(1-rd)$. Then [4] showed that $0 < A \leq T_N \leq B$, for $1 \leq s$; and $B < T_N < A$, for $1 > s$. A and B depend on the h which was used to expand the series, but the series can easily be shifted to a new h^* by multiplying the j -th term by $(h^*/h)^{j-1}$.

We wish to choose an optimal stepsize ratio $U = h^*/a$ as large as possible while maintaining one of the error measures:

- Type i) absolute truncation error per step, $|T_N|$; or
- Type ii) absolute truncation error per unit step, $|T_N/h|$;

less than a prescribed tolerance, ϵ . Relative or mixed error measures are handled in a similar manner. For Type i) error if $s < 1$, then U is the single real root on $[0, 1]$ of

$$p(x) = \frac{V_N}{r^{N-1}} x^N + \epsilon x - \epsilon = 0.$$

Let $\delta = \epsilon r^{N-1}/V_N$. Then $\delta^{1/N}(1 - \frac{1}{N}\delta^{1/N})$ is an approximate root of $p(x)$, so we use it as an estimate for U . Estimates for other error types can be computed similarly, and some are given below. Figure 1 shows an example of how these estimates depend on the tolerance, ϵ , and on the series length, N .

Error Measure	Estimated Optimal Stepsize Ratios.
Type i)	
$s < 1$	$\delta^{1/N} (1 - \frac{1}{N} \delta^{1/N})$
$s \geq 1$	$(\frac{\delta}{d})^{1/N} (1 - \frac{d}{N} \delta^{1/N})$
Type ii)	
$s < 1$	$(a\delta)^{\frac{1}{N-1}} (1 - \frac{1}{N-1} (a\delta)^{\frac{1}{N-1}})$
$s \geq 1$	$(\frac{a\delta}{d})^{\frac{1}{N-1}} (1 - \frac{d}{N-1} (a\delta)^{\frac{1}{N-1}})$

Table 1. Optimal Stepsize Ratios

We have shown how to compute an optimal stepsize ratio. Next we consider the choice of the series length. The codes produced by the various translator programs use recurrence relations to generate the series. Fast series generation techniques have not yet been found suited to this application, so each integration step requires $\alpha N^2 + \beta N + \gamma$ floating point operations. The approximate cost of solving the entire problem is proportional to

$$C(N) = \frac{\alpha N^2 + \beta N + \gamma}{U}$$

As an example, the code produced by Chang's ATS translator for $y'' = y - (y')^2$ requires approximately $1.5N^2 + 15N + 100$ floating point operations per step. The translator programs can be modified to supply their object codes with the values for α , β , and γ so that $C(N)$ can be evaluated at run time to choose an optimal series length, N_{opt} . As experience with other

methods has indicated, problems with a more stringent error tolerance are solved more efficiently by higher order methods (longer series). As indicated by Figure 2, $C(N)$ typically has a broad minimum, so the exact choice of N is not critical. In practice, a series length longer than N_{opt} should be used

because the increased accuracy of the estimates for R_c from a longer series balances the increased cost of its generation.

The preceding discussion assumed that the solution to the ODE had only one primary singularity. If the solution has a conjugate pair (a and \bar{a}) of primary singularities of order s , then its series is asymptotic to ΣF_j , the series

for $f(x) = (a - z)^{-s} + (\bar{a} - z)^{-s}$. If Σv_j is the

series for $v(z) = (|a| - z)^{-s}$, then $F_{j+1} = 2V_{j+1} \cos(s+j)\theta$, where $\theta = \arg a$. Hence the truncation error for f is at most twice the truncation error for v and the estimates for U given in Table 1 for a single primary singularity remain valid for a conjugate pair of primary

singularity remain valid for a conjugate pair of primary singularities, provided that ϵ is replaced by $\epsilon/2$.

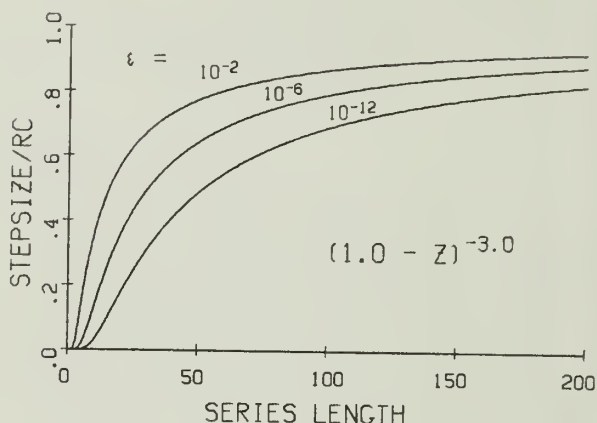


FIGURE 1. OPTIMAL STEP FOR ABSOLUTE TRUNCATION ERROR PER UNIT STEP.

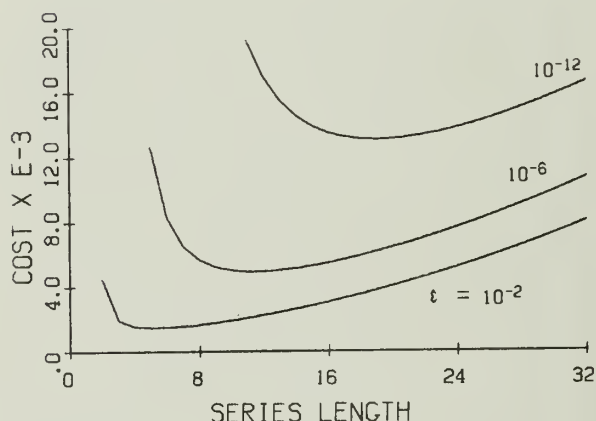


FIGURE 2. COMPUTATIONAL COST FOR $y'' = y - (y')^2$ WITH $\alpha = 1.5$, $\beta = 15$, AND $\gamma = 100$.

We have shown how to choose dynamically a series length which approximately minimizes the computational cost of solving a system of ODE's by series methods. The optimal ratio of the stepsize to the series radius of convergence has been given. These choices depend on series length, locations and orders of primary singularities, error tolerance, and the type of error measure the user wishes to control. Work is in progress with Roy Morris to incorporate the ideas in this paper into a suitable translator program.

REFERENCES

- [1] D. Barton, I.M. Willers, and R.V.M. Zahar, Taylor series methods for ordinary differential equations - An evaluation, Mathematical Software, Edited J. Rice, Academic Press, 1972, pp. 369-390.
- [2] Y.F. Chang, Automatic solution of differential equations Springer-Verlag Lecture Notes #430, 1974, pp. 61-94.
- [3] Y.F. Chang and G. Corliss, Convergence analysis of compound Taylor series, Proc. 8th Manitoba Conf. on Numerical Mathematics and Computing, to appear.
- [4] G. Corliss and D. Lowery, Choosing a stepsize for Taylor series methods, J. Comput. Applied Math., (3) 1977, pp. 251-256.
- [5] J.A. Leavitt, Methods and applications of power series, Math. Comput. 20 (1966), pp. 46-52.

A Semi-Implicit Mid-Point
Rule for Stiff Systems of
Ordinary Differential
Equations

P. Deuflhard (speaker),
G. Bader
University of Heidelberg

In this talk, three versions of a new extrapolation method for stiff systems of ODEs are presented.

Let

$$(1) \quad y' = f(t, y), \quad y(0) = y_0$$

denote the stiff initial value problem. Then the main idea is to rewrite the system in the form

$$(1') \quad y' - Ay = f(t, y) - Ay =: \bar{f}(t, y),$$

with

$$A \approx f_y(0, y_0)$$

and to discretize the above left-hand side separately - but not in terms of a sophisticated approximation of the associated matrix exponential. Rather, in view of the extrapolation to be combined with the discretization, the crude approximations (stepsize h)

$$(2) \quad E(Ah) = I + Ah, \quad E(-Ah)^{-1} = (I - Ah)^{-1}$$

turn out to be sufficient. The main difficulty in the construction of such an extrapolation method is to assure both the existence of an asymptotic h^2 -expansion (that permits one to use h^2 -extrapolation) and desirable stability properties (that make the method efficient for stiff ODEs).

The semi-implicit mid-point rule to be presented here has been derived from applying the explicit mid-point rule (GBS-method) to the equation (1') using the exact matrix exponential $\exp(Ah)$ and replacing it deliberately by either $E(Ah)$ or $E(-Ah)^{-1}$. With (2), the following scheme is obtained (compact version)

(3) a) semi-implicit Euler starting step:

$$\begin{aligned} \eta_0 &:= y_0 \\ \Delta_0 &:= (I - hA)^{-1} h f(t_0, y_0) \end{aligned}$$

b) semi-implicit mid-point steps:

$$\begin{aligned} k &= 1, \dots, \ell-1 : \\ \eta_k &:= \eta_{k-1} + \Delta_{k-1} \\ \Delta_k &:= \Delta_{k-1} + 2(I - hA)^{-1} [h f(t_k, \eta_k) - \Delta_{k-1}] \end{aligned}$$

c) final smoothing step:

$$\begin{aligned} \eta_\ell &:= \eta_{\ell-1} + \Delta_{\ell-1} \\ \Delta_\ell &:= \Delta_{\ell-1} + (I - hA)^{-1} [h f(t_\ell, \eta_\ell) - \Delta_{\ell-1}] \end{aligned}$$

$$S_{\ell} := \eta_{\ell-1} + \Delta_{\ell} .$$

With $S_{\ell} =: y_h(t)$, $\ell h = t$ fixed, the existence of an h^2 -expansion is shown. For stability reasons, $\ell=4m$, m integer, is chosen.

On the basis of algorithm (3) and of detailed investigations in terms of S-stability, two further schemes have been developed. In one of these schemes, the starting step (3a) is replaced by an implicit Euler step. In the other scheme, the smoothing step (3c) is, in addition, repeatedly applied at every t_{2i} . The iterative application of that smoothing step leads to a special realization of the following, fully implicit discretization scheme:

$$\begin{aligned} (4) \quad & \eta_0 := y_0 \\ & \eta_1 := \eta_0 + h f(t_1, \eta_1) \\ & \eta_2 := \eta_1 + h f(t_2, \eta_2) \\ & k = 1, \dots, 2m-1: \\ & \eta_{2k+1} := 2 \eta_{2k} - \eta_{2k-1} \equiv \eta_{2k} + h f(t_{2k}, \eta_{2k}) \\ & \eta_{2k+2} := \eta_{2k+1} + h f(t_{2k+2}, \eta_{2k+2}) . \end{aligned}$$

This scheme may be regarded as an implicit trapezoidal rule on a $2h$ -grid:

$$\eta_{2k+2} - \eta_{2k} = h \left[f(t_{2k}, \eta_{2k}) + f(t_{2k+2}, \eta_{2k+2}) \right]$$

started by two implicit Euler steps on an h -grid. The intermediate quantities η_{2k+1} are only used to construct useful starting points for a Newton-like iteration to solve the implicit step (details omitted here). Moreover, in the implicit steps, an additional stepsize restriction is derived on the basis of the associated affine invariant convergence theorem. Once more, an h^2 -expansion exists.

The three above sketched extrapolation methods have been tested extensively, among other examples also by means of the well-known sample set STIFF DETEST. The results seem to indicate that the semi-implicit midpoint rule in combination with the implicit Euler starting step is the preferable choice. All of the methods compare quite well with the code GEAR (due to Hindmarsh). Detailed comparisons will be given.

WHAT MIGHT ODE SOLVERS BE TESTED FOR?

T.E. Hull
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada

Outline

I have found it convenient to divide the material of this talk into three parts. The first part is intended to provide a brief summary of the testing that took place during the "classical" period of the 1950's and 1960's. The second is intended to provide a similarly brief summary of the "modern" period, the 1970's. The third and more extensive part consists of some speculation about what we might look forward to in a "future" period of testing of ODE solvers.

The emphasis is primarily on testing in the experimental sense, that is, on testing programs for solving ODEs on a computer. However, testing in the theoretical sense is included as well, as for example in testing mathematically that a certain class of formulas is asymptotically unstable, since it seems to me that the two aspects of testing complement one another and are best considered together.

In a relatively brief survey of such a large field, it is impossible to be complete, and in particular to attempt anything like a complete bibliography. However, several recent papers and books do contain substantial lists of references. I therefore decided to omit references altogether from this summary, and only mention in the text itself a number of authors' names and dates. Those that are mentioned are intended to be reasonably representative, but by no means complete.

The next section of this summary is devoted to the first, or classical stage of the 1950-60's. This stage is characterized primarily by a study and comparison of mathematical formulas, as opposed to anything like what we would now call mathematical software. The mathematical results obtained during this period were quite substantial, however, and had very important practical implications, particularly with regard to the stability properties of formulas (first for non-stiff, and later for stiff problems). Experimental testing provided some important results as well, for example in pointing to the possible importance of variable order Adams formulas, but the experiments were relatively modest comparisons of various formulas using fixed step sizes.

Then a further section of this summary is devoted to the second, or modern stage of the 1970's. This stage is characterized by a shift in

emphasis to relatively large scale testing of "real" computer programs over fairly substantial classes of problems. There were also a number of important theoretical contributions related to the testing of various approaches during this period, for example with regard to the stability of methods for stiff systems, but the emphasis in testing had certainly shifted towards large scale experiments. Based on the results obtained, a number of conclusions have been reasonably clearly established about the reliability and efficiency of various classes of computer programs, particularly for non-stiff problems, but also to a limited extent for stiff problems as well. However, the developments and conclusions attributable to this stage brought with them many new and interesting questions about what we should be trying to accomplish when we test ODE solvers.

The final section of this summary is intended to help clarify a few of these questions, and perhaps provide some suggestions about what steps we might take to resolve some of them. It is clear that our objectives must be more clearly defined, that we must broaden the classes of problems over which the tests are performed, and above all that our testing must reflect as closely as possible the real needs of the ultimate user, for example in terms of such factors as convenience, robustness, flexibility, and so on. Such changes will of course reflect back in turn on design objectives for the development of good numerical software for solving ODEs.

Classical stage

As already indicated, the important developments of this stage appear to be more theoretical than experimental. The beautiful results established by Dahlquist in the mid 1950's can be considered, from the point of view of testing, as providing a definitive test for determining which multistep formulas are asymptotically stable and which are not.

Then in 1963 Dahlquist "did it again", when he published a paper on A-stability which provided the impetus for much of the subsequent growth of interest in methods for solving stiff problems. The ideas and the criteria were essentially theoretical. Other theoretical developments were also taking place in the 1960's. With the help of hindsight, we see that among the more notable from the point of view of testing mathematical formulas

for solving ODEs, were the extrapolation method developed by Gragg (and extended by Bulirsch and Stoer) and the Runge-Kutta formula pairs developed by Fehlberg and others.

The experimental testing of this period was relatively limited, and was restricted mostly to comparing formulas of fixed order, using fixed stepsizes, on relatively small numbers of problems. For example, some attempts were made, independently by Ralston and Johnston, to determine which were the better Runge-Kutta formulas. Of the testing I was associated with during this period, I believe that the results obtained with Creemer and published in 1963 were the most significant. After several years of trying to determine the best combination of accuracy and stability in multi-step formulas, we finally obtained experimental evidence that convinced us we should use Adams formulas, and worry about order, and not about more accurate formulas of a particular order. We also concluded that one corrector per step was optimal, but that this corrector needed to be followed by an evaluation (PECE) for stability. We didn't foresee the development of variable order, variable step Adams codes which are so common nowadays, but I wish we had!

Modern stage

During the 1970's there has been a rapid development of techniques for handling stiff systems, including theoretical criteria for testing the stability of various formulas. Theoretical criteria were also developed for non-stiff methods. For example, criteria for variable order, variable step methods were developed at the University of Illinois. More recently a theoretical criterion for testing Runge-Kutta formula pairs was developed at the University of Toronto.

Led by the publication of Gear's DIFSUB, substantial codes began to be widely available during this period. Some of the better known ones in North America are associated, for example with the names of Bulirsch and Stoer, Hindmarsh and Byrne, and Shampine and Watts. But others have been developed both on this continent and in Europe. (For the purposes of this talk, Britain is considered to be a part of Europe.)

Experimental testing of such programs also began to be carried out on a relatively large scale. An early series of tests was carried out at the Bell Telephone Laboratories. Even more extensive testing has been done at the Sandia Corporation, at the University of Manchester, and at the University of Toronto. The emphasis in all of this testing has been primarily concerned with reliability and efficiency, in one form or another, of the various methods.

The Toronto testing program is called DETEST. It has changed considerably over the years, but basically it collects statistics on reliability and efficiency over several classes of problems. Care was taken, in all the ways we thought were possible, to produce results that were not affected appreciably by relatively separate issues, for example by issues such as determining the starting stepsize.

The original design was aimed specifically at

comparing methods, as opposed to simply making independent assessments of them. This caused us to insist that each method try to accomplish the same task, namely to keep the error per unit step below the prescribed tolerance, and this led us to modify the programs accordingly. This caused a great deal of controversy. Moreover, an "error per step" rule is more efficient and, in addition, makes it possible to step over the most common kinds of discontinuities. The present version of DETEST therefore does not require any modification of the program being tested. It is aimed more at the assessment of methods, rather than their comparison. However, a new notion of "normalized statistics" has been introduced to provide an indirect method of comparison. The present version is also much more flexible. For example, it is easy to select what problems are to be used, or to add new ones, and global statistics can be obtained, as well as local ones. NEW DETEST also contains a class of problems with discontinuities.

Out of all the testing that has been done on non-stiff problems, a number of general conclusions have become established. For example, one happy conclusion is that there do not seem to be any serious "class distinctions", by which I mean that methods that are relatively good for one class of problems seem to be relatively good for the other classes as well. A second conclusion, which is to be expected, is that variable order methods are needed to handle a wide range of tolerances. A third and most interesting conclusion is that function evaluations need to be relatively expensive to justify the extra overhead that multi-step methods require in order to keep down the number of function evaluations. This last point was the key one in comparing extrapolation methods with Adams methods according to our earlier tests, while the Runge-Kutta methods were occasionally unreliable. However, the newer Runge-Kutta methods based on formula pairs appear to have displaced the extrapolation methods as the best ones to use when function evaluations are relatively inexpensive.

The corresponding situation with the results of STIFF DETEST is more complicated, and the conclusions are not nearly so definite. For one thing, the Jacobian plays an important role, and an important factor in assessing different methods is the dimension of the problem, since two methods may differ in the relative number of n^3 and n^2 operations they perform. Also it may matter whether the Jacobian is determined numerically, whether it is sparse or not, and so on. Other important factors appear to be the nearness of the eigenvalues to the imaginary axis, the degree of non-linearity, the kind of coupling between the smooth and non-smooth components of the solution, and the tolerance. Also, methods are more likely to fail on stiff problems, and this makes the collecting of meaningful statistics more difficult.

Nevertheless, it does appear that programs based on backward differentiation formulas stand up very well, their main weakness being when the eigenvalues are near the imaginary axis. But there are other good methods as well, and much more needs to be learned about them. For example, there are good methods based on second derivative formulas, implicit Runge-Kutta formulas, extrapolation, and block implicit techniques.

The future

Of course it is easy to see that in the immediate future we can expect further testing in terms of reliability and efficiency of more programs over broader classes of problems. But I would like to make some brief comments on a number of specific areas in which I consider it would be especially worthwhile trying to improve on what we have been doing. One common feature in each area is, I believe, an attempt to focus attention on those aspects of testing that reflect the real needs of the ultimate user of ODE software, at least insofar as it is realistic to do so.

As a first point, consider the interpretation of the tolerance. As a consequence of earlier arguments about error-per-step and error-per-unit-step, a number of us have become convinced that a very good way to interpret the parameter TOL is in terms of proportionality. The design goal for the ODE program is then to produce answers whose global error has a norm that is proportional to TOL. (This is intended to be quite independent of whether the error is relative or absolute, or of whether components of the error vector are weighted in some special way.) The factor of proportionality is of course problem dependent, but this interpretation is easily understood by a user, and also easily exploited by him. The current version of DETEST computes a measure of how smoothly the error depends on TOL, but improved measures need to be considered. It should be noted that it is often particularly difficult to meet this design goal for large TOL. It should also be noted that the frequency with which output is requested could have a considerable effect on how well some programs meet this goal.

A second point that I would like to mention concerns the scale of a problem. I have often thought that it was not unreasonable to expect the user to provide some measure of the scale of his problem, as part of the problem specification. Requiring HMAX is a possibility, but HMAX has the disadvantage that it depends on the method as well as the problem, and the user should only be required to provide information about his problem. Many users would object to any such requirement (even if we had a good idea of what we meant by it!), but I would at least like to have the option of providing a value of SCALE, if only as a "knob" that can be turned one way to make the method more reliable (and the other way to make it more efficient of course). In our earlier tests of Runge-Kutta methods we found some of them to be quite unreliable, until we imposed a maximum on the stepsize, in which case the improvement was dramatic. The test results can therefore depend in a crucial way on how this question is handled. Of course we can simply use default values, or build heuristics into our programs to estimate the scale of a problem. In any event, I believe that more attention should be given to this question. (It should of course be acknowledged that this question is made more complicated by the fact that the scale may change during the course of the integration.)

What other measures of reliability or robustness should we test for? Can we test a program for its ability to detect stiffness, or the presence of discontinuities, or trouble with roundoff? Can we

test separately its ability to get started? Or to cope with discontinuities? What about improper use? (A common example of the latter is undefined input variables, but our programming languages seem to make it virtually impossible to deal with this problem in a decent way.)

With regard to efficiency, it seems to me that more thought should be given to standardizing the way in which we present the results. For example, with non-stiff methods for a particular class of problems, I wonder if we could present the results in terms of a small number of parameters. One possibility might be the parameters h , f , t and k , where the computing cost has been determined to be

$$h + f * F + t * TOL^{-k}.$$

Thus h is a kind of overhead cost, f is the coefficient related to the cost F of evaluating functions, while t and k appear only in the contribution due to TOL. It would be particularly helpful if some such result could be normalized in such a way that it was relatively independent of the problem class. Of course it would also be helpful if the normalization could factor out the dependence on a particular machine, and, as well, the dependence on implementation details such as how the compiler handles subscripts. Other components of cost, such as storage, and other properties of the method, such as how well it gets started, or how well it copes with discontinuities, would also have to be considered, but it would be very helpful if these could also be presented in some sort of normalized way.

The same sort of objective would of course be desirable in the case of stiff systems as well, but the situation would be much more complicated here. Nevertheless, we are already measuring the number of times an n^3 process is invoked, the number of n^2 processes, and so on, so we have some of the ingredients.

I continue to believe that theorems about programs are important. I have in mind, for example, proving that a particular method solves all problems in a particular class, in the sense that it produces $z(x)$ such that $\|z'(x) - f(x, z)\| \leq TOL$. I realize that such theorems can be established only over very limited classes, and are really not very practical. However, I continue to believe that establishing of such theorems is a good necessary condition to be satisfied and, moreover, that the weakness of the conditions under which such a theorem can be established is a measure of the strength of a method.

With regard to broadening the class of problems over which methods are tested, I have a number of points in mind. One has been impressed on me because none of our problems ever uncovered the inability of Fehlberg pairs of order more than 4 to solve simple quadrature problems of the form $y' = f(x)$. A second point is to have more problems in which stiffness changes. A third is to develop problems for testing the sensitivity of methods to changing parameters. (I would like to ask what we should do about the situation that often arises in chemical kinetics when, due to a minor numerical error, it is possible for one of the concentrations to become negative and throw the calculation off

completely. If we do nothing about this difficulty, one method might appear to be much worse than another, purely because of a relatively minor event that should really have a negligible effect on the final results.) Sparseness is another factor that should be taken into account with new classes of problems.

Besides being important to add new classes, we should begin to weed out some old ones. Surely, by now, we should be able to identify some problems that are redundant and can be removed to make room for others.

I would like to conclude with a reminder that good documentation is needed and, if we are testing programs, we should include some sort of assessment of their documentation. Of course I don't mean to suggest that this can be quantified, but some kind of assessment would be helpful. Perhaps this would also involve a statement about how convenient the program is to use, about what options are available (that have not already been considered). One particularly difficult option to deal with is the incorporation of stopping criteria other than the standard one of stopping at a particular value of the independent variable. Another property that must not be forgotten is the extent to which any special knowledge that the user might have can be exploited, regarding the stiffness of the problem for example, or some measure of the scale, or whether the problem is linear or not, or whether the user can provide the Jacobian, and so on.

Altogether, it appears that our ODE solvers might be tested for many different things, and some of those properties that are already being tested might be done in ways that are at least somewhat more carefully defined and useful.

The NAG Library Runge-Kutta Routines

by

Ian Gladwell
University of Manchester
England

Abstract

The latest version of the NAG library includes a suite of subroutines based on the Runge-Kutta-Merson formula. This suite includes a main routine with a variety of features and a number of driver routines designed, in the main, for "black-box" use. These routines and a number of auxiliary ones are described in detail.

1. Introduction

The NAG library ordinary differential equation chapter contains a set of subroutines based on the Runge-Kutta-Merson formula for integrating the initial value problem

$$y' = f(x, y), \quad y(X) \text{ given} \quad (1.1)$$

over a range $[X, XEND]$. At the lowest level there is a subroutine DO2YAF which is designed to take one step of predetermined length using the Merson formula. This will be described in more detail in the next section. At the next level is a general subroutine DO2PAF which is designed to integrate across a range by calling DO2YAF at each step. DO2PAF also has a number of other features such as interrupt and interpolation facilities and these will be described in detail in section 3. At a higher level still there are several driver routines for DO2PAF. In the main, these routines have been designed so that the inexperienced user can solve his problems without recourse to DO2PAF. However, one driver routine, DO2BDF, is designed to calculate global error estimates and to perform stiffness checks, and this is intended for general use. All these routines are discussed in section 4.

Specifications of all the subroutines discussed here can be found in the NAG Manual, Mark 7 (1979).

2. Subroutine DO2PAF

In its normal mode of operation routine DO2YAF takes a step of length H from a current point X at which point the solution is stored in the array Y . It produces the solution at $X + H$ in Y and it reorganises columns of the workspace array to retain values of y' at $X + H$ and y and y' at X for use by the interpolation routines which are to be discussed in section 3 below. The routine DO2YAF also returns a local error estimate in a column of the workspace and in yet another column an indication of whether the

calculation of the error estimate was significant, componentwise. In designing this significance test we observe that the error estimate is calculated by componentwise subtraction of two approximate solutions at $X + H$ (as is almost always the case in Runge-Kutta routines). We flag insignificance if, in this subtraction, all the significant digits of the two solutions cancel to rounding error level. Since the step H is determined (in routine DO2PAF) from the error estimate we feel that it is important to be aware if this determining factor is completely dependent on the wordlength characteristics of the computer in question. In fact, the workspace array has been so arranged that all the workspace used on the intermediate steps in the Runge-Kutta step is used again in returning the information about the solution and its derivatives, and the error estimate and its significance.

There are two other modes of calling routine DO2YAF. Both are designed for use on calls from routine DO2BDF. This routine computes global error estimates for solutions of routine DO2PAF and attempts to determine the degree of "stiffness" of the system (1.1). In neither of these calls is there any need to retain information for interpolation purposes and immediate storage savings are made. Also, when calculating global error estimates the stepsize is controlled by independent calls to DO2PAF so that the local error estimate is not needed, leading to savings in computer time and storage. When calculating the degree of stiffness, one of the algorithms used is essentially that of Shampine (1977) which requires a local error estimate for Euler's method with step H for use in a comparison with the Runge-Kutta-Merson error estimate. This is returned in a column of the workspace used on a normal call for returning interpolation information.

3. Subroutine DO2PAF and auxiliary interpolation routines

Routine DO2PAF is designed to integrate the system (1.1) from an initial point X to a final point $XEND$ using a sequence of calls to routine DO2YAF. The stepsizes in the calls to DO2YAF are designed to control the local error, with a special short last step to reach $XEND$ exactly. In many respects DO2PAF is similar to well-known routines RKF45 (Shampine and Watts, 1976a) and DVERK (Hull et al., 1976). We will only report on the major points of difference between DO2PAF and these routines.

Subroutine DO2PAF has the calling sequence

```
SUBROUTINE DO2PAF(X, XEND, N, Y, CIN, TOL, FCN,
1 COMM, CONST, COUT, W, IW, IW1, IFAIL)
INTEGER N, IW, IW1, IFAIL
REAL X, XEND, Y(N), CIN(6), TOL, COMM(5), CONST(3),
1 COUT(14), W(IW, IW1)
EXTERNAL FCN
```

The parameters X and XEND are discussed above, Y is used to pass the computed solution to and from the routine, TOL is used for local error control, FCN is the usual routine used to define the differential system, the array W is used mainly for workspace, and IFAIL is an error indicator to be discussed below.

There are four arrays CIN, COMM, CONST, COUT used for communication with routine DO2PAF, and in certain circumstances columns of the workspace array W are also used. In general terms, the array CIN is used to communicate information to control the mode of entry to the routine, to determine the type of error control, and to supply initial, maximum, minimum and continuation step-sizes when needed. The array COMM is used to handle a variety of facilities designed to interrupt the integration in different ways. If the integration is controlled largely through the use of interrupts with no intention to integrate to XEND, then the value XEND is essentially used to define a "scale" for the integration. Both CIN and COMM are described in more detail below. The array CONST can be used to pass control constants to the routine to assist in the internal stepsize strategy. Two of the elements of CONST can be used to vary the maximum and minimum ratios in attempted stepsizes on successive steps. (Of course the stepsize can be reduced as much as necessary at any step by trying a sequence of stepsizes of successively smaller magnitude.) The third element of CONST can be used to indicate if the differential system is linear and then a stepsize strategy is adopted to exploit the special properties of the Runge-Kutta-Merson formula for linear problems. The array COUT is used to return information to the user. Some of this information is statistical, for example maximum and minimum stepsizes used so far in the integration, but other elements of COUT are used to assist in interpolating the solution within integration steps, and yet others carry information between interrupts. This eliminates the need for use of COMMON but places the user on trust not to change elements of COUT at interrupts. The elimination of COMMON is important in a library environment as it permits integrating separate systems of differential equations simultaneously using the interrupt facility. The use of the four communication arrays (five including the workspace W) rather than just one, as in DVERK, is designed to permit future development in a library environment without major reprogramming effort.

We now discuss the use of the control array CIN. The element CIN(1) is used to control the mode of integration. On initial entry, CIN(1) must be set to zero or one. If it is set to zero, default values of the other elements of CIN, and the elements of COMM and CONST are used. The routine DO2PAF is designed to treat this choice as a special case and a fast integration from X to

XEND results. If CIN(1) is set to one, all the other elements of CIN and the elements of COMM and CONST must be set (the choice zero giving default values in all cases). On return, by reaching XEND or by interrupt, a value of CIN(1) between two and six is obtained (depending on the mode of exit), and the routine can be re-entered with this value. The choice CIN(2) = 0.0 (the default) gives mixed relative/absolute error control on the maximum norm of the local error. Similarly CIN(2) = 1.0 and 2.0 give absolute and relative (with floor) local error tests respectively, and the choices CIN(2) = 3.0 and 4.0 give componentwise mixed and absolute local error tests respectively (a relative test is a special case of the mixed test in this case). In these latter cases, columns of the workspace array are used to pass componentwise error control information. The element CIN(3) can be used to specify a minimum stepsize. However, a minimum stepsize designed to ensure progress across the range of integration is calculated and used if it is larger than the user's choice. (The minimum stepsize is actually used only to activate error exits.) The element CIN(4) can be used to pass a maximum stepsize, the default being $0.5(XEND - X)$. The element CIN(5) can pass an initial stepsize estimate (and returns the initial stepsize used). If the estimate is smaller than the minimum or larger than the maximum then the corresponding minimum or maximum is used. If a default value is requested, then an estimate approximately proportional to $(\|y\|/\|f\|) * (XEND - X) * (MACHEPS * TOL)^{1/2}$ is used, where p is the order of the method. Safeguards are taken against zero values arising in the norms. This estimate usually gives a step rather smaller than the largest stepsize allowed by the local error control. This is deliberate as an iteration is employed to compute a realistic initial step and care must be taken to avoid overflow in this iteration. The element CIN(6) is used only for output of the next stepsize in any exit from DO2PAF. This stepsize is used on re-entry.

The array COMM is used to enable interrupt control of DO2PAF. If COMM(1) > 0.0, it is used as an upper bound on the maximum norm of the solution as the integration proceeds. The integration is interrupted if this bound is exceeded (this facility is designed to enable users to avoid overflows). If COMM(2) > 0.0 it is used as an upper bound on the number of permitted calls to routine FCN, in the usual way. If COMM(3) > 0.0, an interrupt is made on a change of sign of $Y(I) - Z(I)$, $I = 1, 2, \dots, N$ where the user's chosen values $Z(I)$ are passed in a column of workspace (this facility is useful for root-finding). If COMM(4) < 0.0, an interrupt is made when $X - COMM(5)$ changes sign (this simplifies output at user-selected points). If COMM(4) > 0.0, an interrupt is made after every step in the manner of many other solvers. All the other interrupts can clearly be accommodated by using COMM(4) at the expense of repeated calls to DO2PAF, but we feel this is unjustified. Because of the interpolation process to be described below two steps are taken before permitting an interrupt, on the assumption that the user will wish to interpolate. Hence if an interrupt of the COMM(1) > 0.0 or COMM(3) > 0.0 type is required on the first accepted step we adjust the stepsize so that the interrupt comes on the second step instead.

As has been previously stated, DO2PAF has been designed to permit interpolation to the approximate solution within integration steps. This interpolation process is designed to use the minimum amount of retained information whilst being of at least the same order of accuracy as the integration. To minimise the amount of retained information we use Hermite interpolation on three points giving an accuracy of $O(h^5)$ as against an accuracy of $O(h^4)$ in the integration. This choice clearly entails using solution and derivative information at $X+H$, X and $X-H'$ where H' is the stepsize previous to the current one. Clearly we could save one column of workspace storage by not retaining, say, the derivative at $X-H'$. However, this would mean losing symmetry in the interpolation process and, in any case, the author feels that it is preferable that the integration error dominate the interpolation error so that by varying the local error tolerance TOL one varies the accuracy at all points approximately equally. We could instead have retained solution values at several earlier points but this would complicate the code far more than our choice. Note that the retained information is used passively, that is for interpolation purposes only, and hence should have no effect on the stepsize strategy. This ensures that the algorithm remains one-step in character. However, it is insufficient just to consider the order of the interpolation error - its actual size is important. The interpolation error at a point t is approximately proportional to

$$p(t) = (t - X - H)^2(t - X)^2(t - X + H')^2. \quad (3.1)$$

If we consider $\|p\| = \max_{t \in [X-H', X+H]} |p(t)|$ as a

function of $r = H/H'$ then its minimum is attained when $r = 1$ and it increases monotonically and quickly as r increases taking the value 5 near $r = 2$ and values of over 100 for $r < 5$. The maximum value of $\|p\|$ is attained on the longer of the subintervals $[X-H', X]$ and $[X, X+H]$. We assume that interpolation is generally only required on the step $[X, X+H]$ and therefore the large values of $\|p\|$ imply that we should restrict stepsize increases to bound the increase in interpolation error from one step to the next. We choose an upper bound of $r = 2$ on this evidence. Two subroutines DO2XAF and DO2XBF are provided to perform the interpolation using information retained in Y, W and COUT on exit from DO2PAF. Routine DO2XAF computes an approximate solution at any point in $[X-H', X+H]$, whilst DO2XBF computes a selected component of the solution in the same range. It was felt that to supply routines for these two extreme cases was justifiable but that any more complicated selection of components might be more expensive to calculate than repeated calls to DO2XBF.

It is widely accepted that the error in integrating the system (1.1) should be approximately proportional to TOL. To ensure this is the case, routine DO2PAF, in common with most other modern subroutines, is designed to vary the stepsize to make the local error vary proportionally and smoothly with TOL. One aspect of this strategy is to avoid choosing stepsizes on the evidence of insignificant error estimates (see section 2). These insignificant estimates can

arise in two ways. During an integration, the error estimate is unlikely to become insignificant unless TOL is so small that the stepsize becomes very small. If this occurs an error exit is taken (other subroutines have error exits for the related case that too much accuracy is requested). Alternatively an insignificant error estimate can be returned by DO2YAF during the initial stepsize iteration. This is just taken as a sign that the current stepsize is too small and the stepsize is increased and the iteration proceeds. An indication of the dependence of the local error on TOL is given through two elements of the array COUT. A count is kept of the total number of successful steps and the number using the maximum stepsize. If the second count is large relative to the first then one might infer that this choice of TOL has not controlled the local error.

4. Driver Subroutines

There are five driver subroutines for DO2PAF. The first and simplest, DO2BAF, integrates the system (1.1) across the range $[X, XEND]$. It is intended for very inexperienced users and hence has a particularly simple calling sequence designed to define the problem, input a local error tolerance TOL, designate some workspace and permit error exits. The second routine, DO2BBF, is similar to DO2BAF but implements slightly more general error control (similar to the use of CIN(2) = 0.0, 1.0 or 2.0 in DO2PAF), and permits output at user-selected points. This output is implemented through a user-supplied subroutine OUT. This routine is called at the initial point X by DO2BBF and a parameter of OUT must be changed to the next output point and so on. Hence, when OUT is called at the current output point, it must redefine the output point in addition to performing its printing task. Internally DO2BBF uses the interrupt $COMM(4) < 0.0$ to pass output points to DO2PAF.

There are also two driver routines designed to solve equations posed in terms of the solution, as the integration proceeds. The simpler of these, DO2BGF, solves an equation $y_i(x) = v$ for x and exploits the interrupt $COMM(3) > 0.0$ to locate the position of x approximately. The second, DO2BHF, solves a general equation $g(x, y) = 0$ by locating a change in sign of g using the interrupt $COMM(4) > 0.0$ (interrupt at every step). In both cases the equations are solved to machine precision using the interpolants defined by DO2XBF and DO2XAF respectively and using a secant/bisection rootfinder CO5AZF based on an algorithm by Bus and Dekker (1975). The equations are solved to machine accuracy to put all the emphasis on the accuracy of the integration, so that by varying the local error tolerance TOL we can simply control the accuracy of the solution of the equations.

The last driver routine, DO2BDF, is designed to solve (1.1) whilst producing a global error estimate and a variety of related statistics, and to give an indication whether the system (1.1) is stiff. The global error estimate is calculated using the two-and-one technique discussed in Shampine and Watts (1976b). The actual solution output should be close to that which would be calculated by a call to routine DO2PAF with the same value of TOL (though it is calculated by calls to DO2YAF only). The calculation proceeds by interrupting DO2PAF at each

step and taking two half steps with DO2YAF to calculate an auxiliary solution, the difference between the two solutions giving the estimate. The other statistics produced include the maximum error estimate in magnitude in the integration for each component and information about the monotonic growth of the error estimate and any sign changes in the estimate. The user is permitted to specify a maximum size for the solution and this can be used to prevent overflow during calculation of the various solutions. Overflow is more likely to occur during calculation of the solution using DO2YAF as there is no error control on this solution.

The stiffness check is based on two separate calculations. The first uses the ideas of Shampine (1977) where an algorithm is described for a Runge-Kutta-Fehlberg method. The stiffness check is based on a comparison of stepsizes predicted from the Fehlberg algorithm and Euler's method. The comparison is dependent on certain properties of the absolute stability regions of the two methods. It is fortuitous that the stability region of the Merson algorithm used in DO2PAF are as well-suited to this application as the Fehlberg stability region. The second stability check is based on the simple observation that if a stiff problem is integrated using a non-stiff solver such as DO2PAF, the stepsizes used are, in the main, controlled by the requirement of stability rather than by the local error tolerance TOL. Hence, if separate calls to DO2PAF are made with local error tolerances TOL and $2^p * \text{TOL}$, twice the number of steps should be used in the first case if the stepsize is being controlled by the local error tolerance. We compute a stiffness check by measuring the deviation from this expected behaviour. In our tests we have found these two checks approximately equally reliable (but the second is more expensive to compute and hence the user may opt to avoid it). In fact we have also tested other possible stiffness checks. For example, we have counted the number of unsuccessful steps relative to the number of successful ones for various values of TOL. One can formulate theories about the behaviour of this ratio for stiff systems but unfortunately this behaviour is not restricted to such systems and the ratio does not provide a reliable measure.

REFERENCES

- BUS, J.C.P. & DEKKER, T. (1975) Two Efficient Algorithms with Guaranteed Convergence for Finding a Zero of a Function, TOMS, 1, pp.330-346.
- HULL, T.E., ENRIGHT, W.H. & JACKSON, K.R. (1976) Users Guide to DVERK - A Subroutine for Solving Non-Stiff ODE's. Department of Computer Science, University of Toronto, Tech. Rept. 100.
- NAG Manual, Mark 7 (1979) NAG Central Office, 7 Banbury Road, Oxford.
- SHAMPINE, L.F. (1977) Stiffness and Non-Stiff Differential Equation Solvers II: Detecting Stiffness with Runge-Kutta Methods, TOMS, 3, pp.44-53.
- SHAMPINE, L.F. & WATTS, H.A. (1976a) Practical Solution of Ordinary Differential Equations by Runge-Kutta Methods, Rept. SAND 76-0585, Sandia Labs., Albuquerque, New Mexico.
- SHAMPINE, L.F. & WATTS, H.A. (1976b) Global Error Estimation for Ordinary Differential Equations, TOMS, 2, pp.172-186.

François E. Cellier

Peter J. Moebius

Institute for Automatic Control
The Swiss Federal Institute of Technology Zurich
ETH - Zentrum
CH-8092 Zurich
Switzerland

The aim of this paper is to present methods to improve the robustness of a general purpose run-time system for digital simulation of continuous systems.

This aspect of simulation software robustness is just one of several possible aspects. One could as well discuss the robustness of a language to formulate simulation problems (simulation language) or the robustness of a compiler for such a language (simulation compiler). Such aspects will, however, not be considered in this article.

The run-time system robustness is a very important aspect which has not sufficiently been taken into account in the development of existing simulation software. The aim of this paper is to specify demands rather than to present optimal solutions. Solutions are presented from an engineering point of view. They improve the run-time system robustness to some extent, but better solutions can certainly be found. It is hoped that some of the Numerical Mathematicians attending this Conference may find these problems of interest and may develop better solutions to them in the future. This is one of the main reasons why the authors decided to present this material at a Conference of Numerical Mathematics rather than at a Conference of Simulation Techniques.

I) INTRODUCTION:

A simulation run-time system can be robust in two senses:

a) The user should never be required to provide any kind of information which he does not have at his disposal. He should be able to concentrate on those factors which have to do with the statement of his problem, and should be relieved, as much as possible, of all aspects which have to do with the way his problem is executed on the machine. He should be able to describe his system as easily as possible in terms which are closely related to his common language, but must not be required to provide a step-size for the numerical integration or to specify the integration algorithm to be used.

b) The run-time software itself must be able to check whether the produced time responses are "correct" (within a prescribed tolerance range). The user, normally, has a more or less precise (although often not mathematically formulated) knowledge of the system he is investigating. He has, however, hardly any "insight information" into the tool he is using for that task. He is, usually, very credulous (the obtained results must be correct because the computer displays 14 digits!), and he has no means to judge the correctness of the

produced results. For this reason, it is vital that each algorithm in the system has its own "bell" which rings as soon as it is unable to properly proceed. Under no circumstances are incorrect results allowed to be displayed to the user.

II) AUTOMATED SELECTION OF INTEGRATION ALGORITHMS:

A huge step towards robust simulation software has been taken in the development of step-size controlled integration algorithms. Before these algorithms existed, the user of digital simulation software was required to supply information concerning the step-size to be used -- an information item which he clearly did not have at his disposal. Now, the user can simply provide a tolerance range for the accuracy of the results. This is identical to requesting the user to identify the smallest number in his problem which can be distinguished from zero. This question can certainly be answered by any user, independently of whether he is an expert in Numerical Mathematics or not, since it is closely related to the physics of the problem, and not to the numerical behaviour of the algorithm.

Available simulation software, up to now, usually offers a comprehensive selection of different inte-

gration algorithms. It does, however, not tell the user which would be the most appropriate one for his particular application. In this way, the user is again confronted with making a decision on something he does not really understand. Experience has shown that the majority of the average users always operate with the default integration method implemented in the package which, in most cases, is a Runge-Kutta algorithm of 4th order. Since he does not know what to specify, he simply ignores that question, and after some time of using the software he has even forgotten that the language provides him with the facility to select among different integration algorithms. So far, no integration algorithm could be found which would be able to handle all kinds of problems equally well, and it is more than doubtful whether such an algorithm could be found at all. The user, who does not make use of the facility to select among different integration rules, will, consequently, often waste a lot of computing power. Although much research has been devoted to the development of different integration methods for the different classes of application problems [3,8], the user has, however, no means to easily determine, from the state space description, the problem class to which his particular application belongs. For this reason, the selection of the appropriate integration algorithm should also be automated.

For this purpose, we try to extract features from an application problem during its execution which are supposed to characterize the numerical behaviour of that particular problem as completely as possible. These features are then combined in a feature space in which we can identify specific clusters for which a particular integration method is optimally suited. The proposed methodology for the solution to this problem originates from pattern recognition.

What features may be used for this purpose? A first feature can be associated with the accuracy requirements for the problem. It can be found that the CPU-cost to execute a particular problem depends on the required relative accuracy. Low order algorithms are appropriate for the treatment of systems from the "gray-" and "black box" area where the available data and models are so vague that a precise numerical integration does not make much sense, whereas higher order algorithms are appropriate for the handling of systems form the "white box" area, e.g. from celestial mechanics. Since the user is requested to specify the wanted relative accuracy, this feature can be extracted from the input data.

Multi-step methods require more CPU-time during their initial phase, but are more economic than one-step methods if integration goes on over a longer interval of simulated time. This can be explained by the fact that one-step methods are self-starting whereas multi-step methods need to be initialized. This leads to a second feature. Since the integration has to be restarted after "event times" (instants at which some state trajectories are discontinuous), multi-step integration is in

favour for purely continuous problems or for problems with few event times, whereas one-step integration is appropriate for combined (continuous/discrete) problems where discrete events occur with a high density.

Each integration algorithm has associated with it a domain of numerical stability. The stiffer a particular problem is (the more the different eigenvalues (λ_i) of the Jacobian are separated), the smaller the step-size (h) must be in order to keep all ($\lambda_i \cdot h$) within the stability region of the algorithm.¹ Fortunately, special integration algorithms could be found for which this restriction no longer holds (A-stability, stiff stability). If such an algorithm is used, the step-size need not be reduced due to restrictions imposed by stability demands, but is determined exclusively by the requirements of accuracy. For this reason the eigenvalue distribution of the Jacobian determines a third feature which must influence our decision as to which integration algorithm to use for the execution of a particular problem.

These three features can now be combined to a feature space as depicted in fig.1.

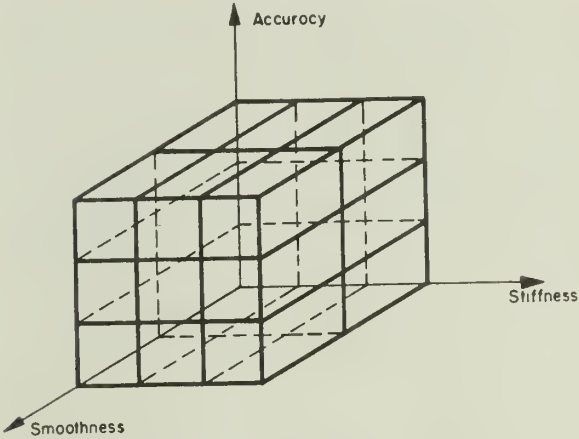


Fig.1: Feature space for selection of integration algorithm

18 clusters have been distinguished in fig.1, and fig.2a and 2b show integration rules which can be associated with them.

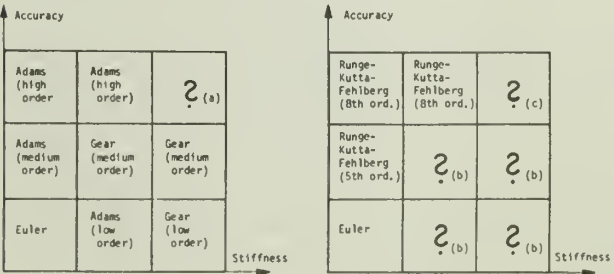


Fig.2: Integration algorithms to be used for smooth (left) and non-smooth (right) problems

As can be seen from fig.2, some of the assignments are still open.

Concerning (a): A Gear algorithm [3] would be appropriate, but it should be of about 8th order (at least for a CDC 6000 series installation -- this depends on the length of the mantissa), whereas, in the Kahaner implementation [6] we use, we have only up to 5th order available.

Concerning (b): For these clusters, a one-step algorithm with a stiff stability behaviour would be most suitable. So far, we have experimented with IMPEX-2 [9], and with DIRK [1], but the programming style of these algorithms, as they are at our disposal at the moment, is not sufficiently elaborate to allow for a fair comparison with the extremely careful and sophisticated Kahaner implementation of the Gear algorithm.

Concerning (c): For this cluster, a high-order stiffly-stable one-step method would be most appropriate. Such an algorithm is, however, unknown to date.

As a matter of fact, the feature space as depicted in fig.1 is still a simplification. To show this, let us consider a system with complex dominant poles close to the imaginary axis. For the treatment of such a system, a stiffly-stable method cannot be properly applied. The system has, however, fast transients, making a Runge-Kutta algorithm not suitable either. Thus, these types of systems, which are called "highly oscillatory" systems, will again require special methods (like stroboscopic methods) for efficient handling (C.W.Gear: private communication). This establishes a fourth feature which is to be used for the determination of the integration method. The reason for the primary simplification lies in the fact that a 3-dim. feature space can be graphed easier than a 4-dim. one (!).

The information provided by these four features is sufficient to determine the best suited integration algorithm for most application problems.

There are even two more features which can be extracted from the eigenvalue distribution of the Jacobian. These are used for other purposes, and will be presented in due course.

So far we have defined features, and we have associated integration methods with them. It remains to determine how these features can be extracted from the state space description of the problem. The first feature (relative accuracy) is user specified on data input. The second feature (smoothness) could also easily be user provided. It is, however, a simple task to detect automatically whether a problem is continuous or combined. If the problem turns out to be combined, one can count the number of event times during a certain period of simulated time, and decide then whether it is a smooth or a non-smooth combined problem. Concerning the third and fourth features (stiffness / highly oscillatory behaviour) one has to compute the

Jacobian out of the state space description of the problem. This can either be numerically approximated at run-time, or one can compute it algebraically by means of formulae manipulation at compile-time. This is rarely done by available simulation compilers but it is feasible, and seems to be a promising approach. The wanted features can now be computed by estimating the critical eigenvalues. The eigenvalues with the largest and smallest absolute values can be approximated by appropriate matrix norms, whereas the real part of dominant poles can be found by estimating the "margin of stability" [5,7]. However, since several quantities are needed, we found that it is in most cases faster to compute the whole set of eigenvalues directly by use of the EISPACK software [2]. The required CPU-time turned usually out to be neglectable compared to the time spent for numerical integration.

III) ADAPTIVE SELECTION OF INTEGRATION ALGORITHMS:

In a nonlinear case, the Jacobian will usually be time dependent, and, with it, also its eigenvalues. Since the classification is, in general, defined for linear systems, it may well be that in a nonlinear case it would be best to assign the integration algorithm dynamically to the problem. For this purpose, one has to recompute the eigenvalues from time to time to find out whether the integration method in use is still appropriate. It seems a good idea to recompute the eigenvalues as soon as the step-size, which is controlled by the integration rule, has changed by an order of magnitude, but not before a minimum time span of maybe 0.01 times the run length has elapsed. This can then be used to obtain an adaptive selection of the appropriate integration scheme.

IV) VERIFICATION OF SIMULATION WITH RESPECT TO MODELING:

Let us assume that a valid model has been derived from the physical system under investigation, and let us question what assurance we have that the time responses which we obtain through simulation represent the (valid) model correctly.

For a variable-step integration method being used, we normally trust in the step-size control mechanism which is equivalent to confiding in the error estimation procedure. This will usually be justified as long as the local error which we control can be used as a valid estimate for the global error in which we are interested.

Experience has shown that local errors will not usually accumulate as long as the system is numerically stable. In a nonlinear case, it may, however, happen that some eigenvalues "walk" into the right half-plane for a short period of simulated time. Let us consider, as an example, the

well known Van-der-Pol equation. Fig.3 shows how the eigenvalues "walk around" during one limit cycle.

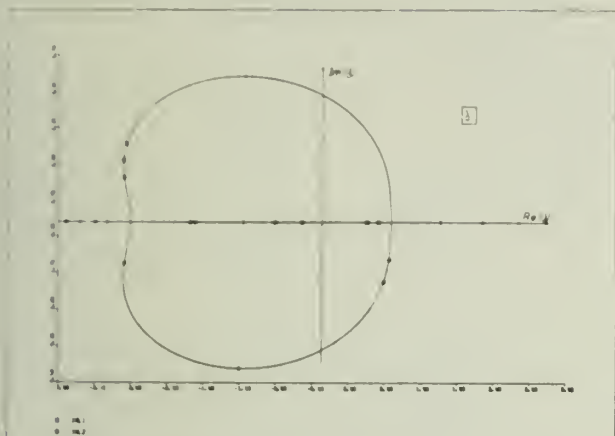


Fig.3: Eigenvalue movement of the Van-der-Pol equation during one limit cycle

As can be seen, the system becomes periodically unstable. During such time intervals, errors will accumulate, and, consequently, we must be careful in the interpretation of the obtained results. This fact should be reported by the software to the user.

For this purpose, we define another feature (stability). A variable STAB is set equal to zero when all eigenvalues lie in the left half-plane and is equal to one as soon as at least one of the eigenvalues moves into the right half-plane.

$$STAB = \begin{cases} 0.0 : \text{Re}\{\lambda_i\} < 0.0 ; i=1, \dots, n \\ 1.0 : \text{otherwise} \end{cases}$$

We now collect statistics on STAB as for time-persistent variables. In this way we obtain the integral of STAB over time divided by the run length:

$$FF5 = \frac{1.0}{|t_{fin} - t_{beg}|} * \int_{t_{beg}}^{t_{fin}} (STAB) dt$$

The fifth feature (FF5) is a real number between 0.0 and 1.0. If it is close to 0.0, the results obtained by simulation have a good chance to be reliable. If it is close to 1.0, the obtained results are most probably nonsense, and they must be cautiously verified.

V) VALIDATION OF THE MODEL WITH RESPECT TO THE SYSTEM UNDER INVESTIGATION:

Another non-trivial question is whether a model,

for given experimental conditions, properly represents the system under investigation. There exist several possible means to help the user in this respect (like asking him to supply dimensions for all variables in the system to enable an automated dimensional analysis). In this section we want to show that the eigenvalue distribution can also help to answer this question to some extent.

It has been shown in [4] that only those eigenvalues of a matrix can be properly computed which fulfil the following inequality:

$$|\lambda_i| > \sigma_1 * \epsilon^{(1/n)}$$

where σ_1 is the largest singular value of the matrix, ϵ is the machine resolution (e.g. $\sim 10^{-14}$ on a CDC 6000 series installation), and n is equal to the order of the model. For higher order models $\epsilon^{1/n}$ approaches 1.0 and hardly any eigenvalues will then be properly computable. Smaller eigenvalues can take any value and small modifications of the elements of the matrix can place them almost anywhere within the band of uncertainty.

If we now assume that the matrix under investigation is a Jacobian of a state space description for a real physical process, then the elements of the Jacobian are extracted from measurements, and cannot be computed more accurately than $\hat{\epsilon}$, which is a relative accuracy of measurement. We, therefore, must assume that, within that relative accuracy $\hat{\epsilon}$, the elements of the matrix can take any values. In this case, we must also assume that eigenvalues of the Jacobian which do not fulfil the more stringent inequality:

$$|\lambda_i| > \sigma_1 * \hat{\epsilon}^{(1/n)}$$

can take any value within that broader band, although they can be much more accurately computed as soon as any particular values have been assigned to all elements of the Jacobian. This means that as soon as there exist eigenvalues for which the second (more stringent) inequality does not hold, small variations in the systems parameters which lie within the inaccuracy of the measurement can make the model non-stiff or stiff or even unstable. Physically seen, these eigenvalues correspond to merely constant modes which could as well be eliminated from the equation set resulting in a model reduction. Numerically seen, these eigenvalues can lead to accumulation of errors so that these modes can drift away over a longer span of simulated time, again resulting in incorrect simulation trajectories.

Together with the eigenvalues, we compute the following quantity:

$$BORD = \sigma_1 * \hat{\epsilon}^{(1/n)}$$

and the number k indicating those eigenvalues whose absolute value is smaller than BORD:

$$k = \sum_{i=1}^n (j_i) ; j_i = \begin{cases} 0: |\lambda_i| \geq \text{BORD} \\ 1: |\lambda_i| < \text{BORD} \end{cases}$$

k represents an integer between 0 and n.

We now collect statistics on the quantity (k/n) as for time-persistent variables, and obtain a sixth feature:

$$\text{FF6} = \frac{1.0}{|t_{\text{fin}} - t_{\text{beg}}|} * \int_{t_{\text{beg}}}^{t_{\text{fin}}} (k/n) dt$$

Also the sixth feature (FF6) is a real number between 0.0 and 1.0. If it is close to 0.0, the model has some chance to be valid. If it is close to 1.0, the model is most probably invalid, and it should be further investigated.

Evaluation of features FF5 and FF6 requires computation of the eigenvalues of the Jacobian once per integration step. Since this can be expensive, it should not be done automatically, but the user must have a switch at his disposal to turn computation on and off. In this way he can use these features during the development of a new model, whereas he can turn computation off during production runs.

VI) DETERMINATION OF CRITICAL STATES:

In section V we have discussed the case where single eigenvalues were situated close to the imaginary axis, and we have seen that in such a case it may be possible to reduce the order of the model.

It is, however, as interesting to discuss the opposite case where single eigenvalues are located in the λ -plane far to the left. We call these modes the "critical states" of the system. Very often one is not really interested in these fast transients. In such a case one could eliminate these modes from the equation set. If the fast transients are important one could at least try to utilize special integration techniques (like using singular perturbations) to expedite integration.

One can, of course, again compute the eigenvalue distribution for the solution of this problem. However, it is not always easy to see which state equations are responsible for such an eigenvalue. For this reason we recommend the following procedure.

We reserve an integer array i_n of length n which is initialized to zero. Each time an integration step has to be rejected due to accuracy requirements not being met, we add 1 to each element of the array $i_n(k)$ for which the accuracy is not met. This implies, of course, that the local truncation error is estimated for all state variables independently.

At the end of the simulation run we divide each element of the array by the total number of rejected integration steps and obtain in this way another set of n real numbers between 0.0 and 1.0. Elements with the largest value indicate critical states.

REFERENCES:

- [1] R.Alexander: (1977) "Diagonally Implicit Runge-Kutta Methods for Stiff O.D.E.'s". SIAM Journal on Numerical Analysis, vol. 14, no. 6 : December 1977; pp. 1006 - 1021.
- [2] B.S.Garbow, Boyle J.M., Dongarra J.J., Moler C.B.: (1977) "Matrix Eigensystems Routines - EISPACK Guide Extension". Springer Verlag, Lecture Notes in Computer Science, vol. 51.
- [3] C.W.Gear: (1971) "Numerical Initial Value Problems in Ordinary Differential Equations". Prentice Hall, Series in Automatic Computation.
- [4] G.H.Golub, Wilkinson J.H.: (1976) "Ill-Conditioned Eigensystems and the Computation of the Jordan Canonical Form". SIAM Review, vol. 18, no. 4 : October 1976; pp. 578 - 619.
- [5] P.Henrici: (1970) "Upper Bounds for the Abscissa of Stability of a Stable Polynomial". SIAM Journal on Numerical Analysis, vol. 7, no. 4 : December 1970; pp. 538 - 544.
- [6] D.Kahaner: (1977) "A New Implementation of the Gear Algorithm for Stiff Systems". Unpublished private communication. For further detail contact: Dr. David Kahaner, University of California, LosAlamos Scientific Research Laboratory, Contract W-7405-ENG-36, P.O.Box 1663, LosAlamos NM 87545, U.S.A..
- [7] M.Mansour, Jury E.I., Chapparo L.F.: (1978) "Estimation of the Margin of Stability for Linear Continuous and Discrete Systems". Internal Report no. 78-01. To be ordered from: Institute for Automatic Control, The Swiss Federal Institute of Technology Zurich, ETH - Zentrum, CH-8092 Zurich, Switzerland.
- [8] J.D.Lambert: (1973) "Computational Methods in Ordinary Differential Equations". John Wiley.
- [9] B.Lindberg: (1973) "IMPEX 2 - A Procedure for Solution of Systems of Stiff Ordinary Differential Equations". Report TRITA-NA-7303. To be ordered from: The Royal Institute of Technology, Stockholm, Sweden.

THE A-STABILITY OF EXPONENTIALLY FITTED LINEAR MULTI-STEP
METHODS EXACT ON PERIODIC FUNCTIONS

Tsvi Bar-David (White)
Lockheed Missiles & Space Company, Inc.

ABSTRACT

A linear multi-step (LMS) method for solving ordinary differential equations (ODE's)

$$\dot{y} = \frac{dy}{dt} = f(y, t), \quad y(t_0) = y_0$$

$$f(\cdot, t) : \mathbb{R}^m \longrightarrow \mathbb{R}^m$$

produces a numerical solution $u_n(h) \approx y(t_n)$ via the recursion

$$\sum_{j=0}^k a_j u_{n+j} = h \sum_{j=0}^k b_j f_{n+j}$$

$$f_{n+j} = f(u_{n+j}, t_{n+j}), \quad t_{n+j} = t_0 + h(n+j).$$

If $a_k \neq 0$ it is called a k -step method. A k -step method is said to integrate exactly a set F of scalar functions if, whenever it is applied to an ODE whose solution is a linear combination of functions in F , it produces a numerical solution equal to the ODE solution at the mesh points $t_n = t_0 + nh$. A k -step method is called exponentially fitted to (the complex number) λ of the order p if it integrates exactly $P(t)e^{\lambda t}$ for all polynomials $P(t)$ of degree $\leq p$.

A k -step method may be fitted to a set of distinct λ 's of different orders. A standard k -step method of order p is nothing more than an exponentially-fitted method fitted to (and only to) $\lambda = 0$ of order p .

A k -step method is called A-stable if, whenever it is applied to an ODE of the form

$$\dot{y} = \lambda y \quad \text{with} \quad \operatorname{Re} \lambda < 0,$$

it produces a numerical solution $u_n(h)$ which for

each fixed stepsize $h > 0$ is bounded for all n . We prove that if a k -step method is exponentially fitted to $\pm i\omega$ (ω real) of order $p \geq 0$ and is A-stable, then the method is implicit and $p \leq 2$. This extends the well-known result of Dahlquist (1963) which states that if a standard k -step method of order p is A-stable, then it is implicit and $p \leq 2$.

We show that if an exponentially-fitted method is exact on (and only on) the functions

$$1, e^{\lambda_1 t}, \dots, e^{\lambda_p t}$$

(the λ_j need not be distinct) then its local truncation error τ satisfies

$$h\tau = c h^{p+1} \pi y_n + O(h^{p+2})$$

$$\pi y(t) = D(D - \lambda_1) \dots (D - \lambda_p)y(t),$$

$$D = \frac{d}{dt}.$$

We then prove that the usual Milne device can be used to estimate the local truncation error of an exponentially-fitted predictor-corrector pair whenever both the predictor and the corrector are exact on the same set of functions.

The Linear Algebra of Discretisation Methods

S. McKee

Computing Laboratory
and Hertford College,
Oxford

This paper presents a number of results in matrix algebra which are of practical value in studying the numerical stability and rates of convergence of discretisation methods for first or second kind singular or non-singular Volterra integral equations, singular or non-singular integro-differential equations and ordinary or delay differential equations.

To illustrate the motivation behind this work let us consider the example:

$$y'(t) = f(t, y(t))$$

$$y(0) = \eta$$

where $f(t, y(t))$ is a real continuous function on $0 \leq t \leq T$, $-\infty < y < \infty$, satisfying

$$|f(t, y(t)) - f(t, y^*(t))| \leq L |y(t) - y^*(t)|$$

for all points

$$y(t), y^*(t) \in [0, T].$$

Let $h \in (0, h_0]$, $h_0 > 0$ and let k, m, n , and N be positive integers such that $N = (k+n)m$ and $(k+n)h = T$. Define the partition $0 = t_0 \leq t_1 \leq t_2 \leq \dots \leq t_N = T$ so that $t_{(i+1)m} = t_{im} + h$, $i = 0, 1, \dots, n+k-1$. Let y_i and $f_i = f(t_i, y_i)$ denote approximations to $y(t_i)$ and $f(t_i, y(t_i))$ and let $y = (y_0, y_1, \dots, y_N)^T$.

Consider the discretisation method

$$\Phi_h y = 0$$

where

$$\Phi_h: \mathbb{R}^{N+1} \rightarrow \mathbb{R}^{N+1}$$

defined by

$$[\Phi_h y]_i = y_i - y_i$$

for $i = 0, 1, \dots, mk-1$, and

$$\sum_{j=0}^{mk} \alpha_j^{(v)} y_{i+j-mk-h} - \sum_{j=0}^{mk} \beta_j^{(v)} f_{i+j-mk}, \quad (1)$$

for $i = mk, mk+1, \dots, N$.

Here the y_i , $i = 0, 1, \dots, mk-1$ are given initial values (usually only specified

for $i = 0, m, \dots, m(k-1)$) and v is a positive integer such that $v \equiv (i-mk+1) \pmod{m}$. The coefficients are real and we take

$$\alpha_{mk}^{(v)} = 1 \text{ for all } v = 1, 2, \dots, m.$$

In matrix notation this yields

$$\Phi_h y = A_h y - h B_h f - g \quad (2)$$

where

$$f = (f_0, f_1, \dots, f_N)^T$$

$$\text{and } g = (\tilde{y}_0, \tilde{y}_1, \dots, \tilde{y}_{mk-1}, 0, \dots, 0)^T.$$

The matrix A_h has the structure

$$A_h = \begin{pmatrix} E & & & & \\ & A & & & \\ & & A & & \\ & & & \ddots & \\ & & & & A \end{pmatrix} \quad (N+1) \times (N+1) \quad (3)$$

where

and B_h has a similar structure, and the order of E is independent of N .

Dahlquist stability can then be interpreted as requiring that $\exists M$, independent of h (and N), such that

$$\|A_h^{-1}\|_{\infty} < M/h \quad (4)$$

and absolute stability by requiring that $\exists M'$, independent of N , such that

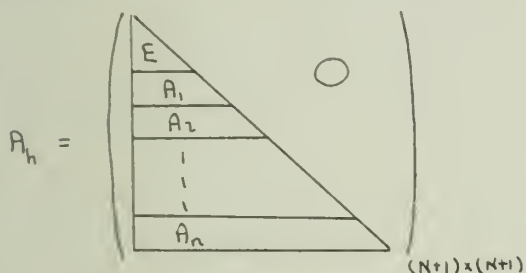
$$\|(A_h - \bar{h} B_h)^{-1}\|_{\infty} < M' \quad (5)$$

where $\bar{h} = h\lambda$ and Φ_h has been applied to the usual test equation $y' = \lambda y$.

It is important to have a more pragmatic characterisation of (4) and (5) which is usually presented in the form of a root condition. This paper is concerned with many different possible characterisations.

The formulation presented here (even though it is only an example) is quite general and includes predictor-corrector (hybrid or otherwise) in any mode, extrapolation (but not the G.B.S. method), Runge-Kutta, cyclic methods, and with a small modification advanced linear multistep methods.

Of course in general A_h and B_h will not be band lower triangular matrices but will be simply lower triangular matrices, e.g.

$$A_h = \begin{pmatrix} E & & & & \\ & A_1 & & & \\ & & A_2 & & \\ & & & \ddots & \\ & & & & A_n \end{pmatrix} \quad (N+1) \times (N+1)$$


For example this arises when product integration techniques are applied to solve the Abel's equation (Holyhead and McKee). Thus this paper includes a study of A_h in this more general formulation.

Dahlquist stability and absolute stability, as defined by (4) and (5), can be characterised by requiring that the eigenvalues of a product of Frobenius matrices all lie inside or on the unit circle, those on having linear elementary divisors. They can be characterised by the approach of Donelson and Hansen (1971), and by the generating function approach of Holyhead and McKee (1976). This paper demonstrates the precise equivalence of these and other different characterisations.

These results have evolved from and in parallel with earlier works by the author and his co-workers. A list of the more pertinent references can be found in the bibliography.

Bibliography

- Allen, K and McKee, S. Discretisation methods for delay differential equations (in preparation).
- Andrade, Celia and McKee, S. On optimal high accuracy linear multistep methods for first kind Volterra integral equations, BIT (to appear).

- Donelson, J and Hansen, E. (1971). Cyclic composite multistep predictor-corrector methods, SINUM, 8, No. 1, 137-157.
- Holyhead, P.A.W., McKee, S. and Taylor, P.J. (1975). Multistep methods for solving linear Volterra integral equations of the first kind, SINUM 12, No. 5, 698-711.
- Holyhead, P.A.W. and McKee, S. (1976). Stability and convergence of multistep methods for linear Volterra integral equations of the first kind, SINUM 13, No. 2, 269-292.
- Holyhead, P.A.W. and McKee, S. Linear multistep methods for the generalised Abel's equation (submitted to SINUM).
- McKee, S. Cyclic multistep methods for solving Volterra integro-differential equations, SINUM (to appear).
- McKee, S. Best convergence rates for linear multistep methods for Volterra first kind equations, Computing (to appear).
- McKee, S. The analysis of a variable step, variable coefficient linear multistep method for solving singular integro-differential equations arising from the diffusion of discrete particles in a turbulent fluid. JIMA (to appear).
- McKee, S. The linear algebra of discretization methods (in preparation).

Acknowledgements

The author is the C.E.G.B. Research Fellow in numerical analysis at Oxford and is grateful for their financial support.

Some Experiments with STRIDE

Fred Chipman
Acadia University

0. Introduction STRIDE is an integrator for systems of ordinary differential equations, designed by Kevin Burrage, J.C. Butcher and Fred Chipman. In section 1, a brief description of the method will be presented, followed by a few details of the algorithm. Section 2 presents preliminary results of experiments with STRIDE on two problems. These results seem sufficiently promising to justify further work on this type of implicit Runge-Kutta method.

1. Description of the method The algorithm implemented in STRIDE is based on the singly-implicit methods introduced in Burrage [3] and Butcher [5]. These methods are of the implicit Runge-Kutta type but with coefficient matrix so chosen as to have only a single distinct eigenvalue. When the method of implementation described in Butcher [4] is used, methods with coefficient matrices of this type require a similar number of arithmetic operations for the performance of its LU factorization as do linear multistep methods.

In the usual notation for Runge-Kutta methods, the stages are given by

$$(1.1) \quad Y_i = y_{n-1} + h \sum_{j=1}^s a_{ij} f(x_{n-1} + hc_j, Y_j) \\ (i=1, 2, \dots, s)$$

and the result at the end of a step by

$$(1.2) \quad y_n = y_{n-1} + h \sum_{j=1}^s b_j f(x_{n-1} + hc_j, Y_j).$$

Let ξ_1, \dots, ξ_s represent distinct zeros of $L_s(z)$, the degree s Laguerre polynomial. Then, with λ any positive number, the coefficients for the particular methods we are considering are given by

$$c_i = \lambda \xi_i, \quad i=1, \dots, s,$$

$$A = (a_{ij}) = CV^{-1}, \quad \text{and } b^T = (1, \dots, 1/s)V^{-1},$$

where $V = (\xi_i^{j-1})$ and $C = (c_i^{j-1}/j)$.

The coefficient matrix A , so defined,

has characteristic polynomial $(z-\lambda)^s$, and, as shown in Butcher [5], can be written

$$T^{-1}AT = \lambda \begin{pmatrix} 1 & \dots & 0 \\ -1 & \dots & \dots \\ 0 & \dots & -1 \end{pmatrix}$$

where, for $T = (t_{ij})$ and $T^{-1} = (t_{ij}^{-1})$,

$$t_{ij} = L_j(\xi_i) \quad \text{and} \quad t_{ij}^{-1} = \frac{\xi_j L_i(\xi_j)}{s^2 L_s^2(\xi_j)}.$$

If $Z_i = Y_i - y_{n-1}$, $F_i = f(x_{n-1} + hc_i, Y_i)$,

($i=1, 2, \dots, s$) and \bar{Z}_i , \bar{F}_i denote the corresponding transformed quantities given by

$$(1.3) \quad \bar{Z}_i = \sum_{j=1}^s t_{ij}^{-1} Z_j, \quad \bar{F}_i = \sum_{j=1}^s t_{ij}^{-1} F_j,$$

and if $H = h\lambda$, then (1.1) is equivalent to the system

$$(1.4) \quad \begin{aligned} \bar{Z}_1 &= H\bar{F}_1 \\ \bar{Z}_i &= H\bar{F}_i - H\bar{F}_{i-1} \quad (i=2, \dots, s) \end{aligned}$$

and (1.2) is equivalent to

$$(1.5) \quad y_n = y_{n-1} - \theta \sum_{i=1}^s \frac{1}{i} L_i'(\theta) H\bar{F}_i, \quad \text{where } \theta = 1/\lambda.$$

In the special case that $\theta = \xi_i$ for some i , and hence $c_i = 1$, the computed result y_n is equal to Y_i , and the method is of order s .

This choice is the one made in STRIDE and, in consequence, the stability properties of the methods are acceptable for many stiff problems. Since the expansion of the solution given by (1.5) is equally valid for other choices of θ , this formula serves as a basis for interpolation of output values and for the provision of starting values for the iterative evaluation of Y_1, Y_2, \dots, Y_s . For stiff problems,

(1.4) is solved using a modified Newton iteration. This requires the LU decomposition of the matrix $(I - HJ)$, where J denotes the Jacobian matrix for the

differential equation. As in most algorithms, J is not computed in each iteration, but only when:

- (a) the convergence rate is unacceptable,
- (b) H has changed by a factor greater than 2 or less than .5 since the last evaluation,
- (c) 20 steps have elapsed since the last evaluation.

This iterative scheme is modified in STRIDE by the use of a relaxation factor $R=2/(1+H/H')$, where H' is the value of H at which $(I-HJ)$ was last evaluated and factored. The rationale behind this modification is that, if μ is an eigenvalue of J then the corresponding eigenvalue of an iterative scheme based on $R(I-HJ)^{-1}$ is $1-R(1-H\mu)/(1-H'\mu)$, and the supremum of this for all μ in the negative half-plane is minimized when R is given the value we use. This minimum value is in fact equal to $|(H-H')/(H+H')| < 1$ so that, for slowly varying Jacobians, eventual convergence would be achieved no matter how much H varies.

For non-stiff problems (1.4) is solved using a fixed point iteration, applied either a fixed number of times or iterated to convergence.

Error estimates are provided by embedding the s-stage (order s) method in an s+1 stage (order s+1) method. As a bonus, error estimates are also provided for methods with orders from 1 up to s (and, if H has remained unchanged for two steps, up to s+1). Thus we are able to implement the methods in a variable order, variable step size manner.

A complete description of the algorithm STRIDE is given in [1], while the theoretical aspects are presented in [2]. Only a few basic parameter descriptions will be given here. The procedure STRIDE has only the four arguments N, FN, JAC and VALUES. N is the dimension of the problem, FN and JAC are procedures for the evaluation of $f(x,y)$ and $J(x,y)$, and VALUES is a user defined procedure by way of which all information is exchanged between the method and the user. Amongst the parameters of VALUES is the integer variable METHOD. The three digits $n_1n_2n_3$ of METHOD indicate to STRIDE the following:

- $n_1=1$ - error per step is to be controlled
- $=0$ - error per unit step is to be controlled
- $n_2=3$ - Newton iteration with an internally approximated Jacobian
- $=2$ - Newton iteration with a user supplied Jacobian
- $=1$ - Fixed point iteration, iterated to convergence
- $=0$ - fixed point iteration applied a fixed number of times.
- n_3 indicates, for $n_2 \neq 0$, the maximum number

of iterations permitted whereas, for $n_3=0$, it indicates the number of iterations to take place.

Finally, in incorporating the increments to both X and Y, the effect of round off error is reduced by computing residuals equal to the intended increments minus the actual changes in the quantities being updated. These residuals are added to the increment for incorporation into the following step. Hence, if ΔY is the increment in Y, we perform the following steps in updating Y:

```
RES:=RES+ΔY;
TEMP:=Y;Y:=Y+RES;
RES:=RES-(Y-TEMP);
```

2. The Experiments In this section we describe several experiments performed with STRIDE using the following problems.

Problem 1: The equations of motion for a satellite moving under the influence of the earth and the moon in a coordinate system rotating so as to keep the positions of the earth and the moon fixed [6].

$$y_1' = 2y_2 + y_1 - u'(y_1 + u)/r_1^3 - u(y_1 - u')/r_2^3$$

$$y_2' = -2y_1' + y_2 - u'y_2/r_1^3 - uy_2/r_2^3$$

$$y_1(0) = 1.2, y_1'(0) = y_2(0) = 0,$$

$$y_2'(0) = -1.0493575098... \text{ where}$$

$$r_1 = [(y_1 + u)^2 + y_2^2]^{1/2}, r_2 = [(y_1 - u')^2 + y_2^2]^{1/2}$$

$$u = 1/82.45 \text{ and } u' = 1 - u.$$

These initial conditions produce a closed orbit with period $T=6.1921693313...$

The solution is required over one period.

Problem 2: A Chemical kinetics problem [7, pp 268-269]

$$y_1' = -.04y_1 + 10^4 y_2 y_3$$

$$y_2' = .04y_1 - 10^4 y_2 y_3 - 3 \cdot 10^7 y_2^2$$

$$y_3' = 3 \cdot 10^7 y_2^2$$

$$\text{with } y_1(0)=1, y_2(0) = y_3(0) = 0.$$

Results are required at $t=4 \cdot 10^k$, $k=0,1,...,9$.

The first problem is non-stiff, and is a good test of the stepsize control. The second problem is stiff, and is frequently used as an illustrative example.

Experiment 1: To investigate the relative efficiencies of the non-stiff options, problem 1 was solved using METHOD = 101, 102, 112, 113 with $EPS = 10^{-5}, 10^{-8}$. The number of function evaluations required were

METHOD =	101	102	112	113	DVDG
EPS=10 ⁻⁵	1070	1882	1882	1882	846
10 ⁻⁶	1687	3165	3548	3193	1288

These results indicate that for this problem there is little use in doing more than one fixed point iteration. This has proved to be the case for all other non-stiff problems that have been tested, and thus METHOD=101 would seem to be a natural choice for such problems. It should be noted however that this method required considerably more evaluations than Krough's DVDG.

Experiment 2: To test the effectiveness of the accumulation of residuals to minimize round off error, problem 2 was solved with METHOD 101, and a relative error tolerance of 10⁻⁸. The results, at the end of one period were:

	y ₁	y ₂	y ₃	y ₄
R	1.2000007	3.08(-6)	-2.58(-6)	-1.0493583
R	1.2000012	4.48(-6)	-2.60(-6)	-1.0493587

The results show a slight improvement with the accumulation of residuals (R).

Experiment 3: To compare the effects of restricting the Newton iteration to a maximum of 2, 3, or 4 iterations, problem 2 was solved with METHOD=122, 123, 124 and EPS=10⁻⁶. The following are the number of function and Jacobian evaluations at t=4x10⁹:

METHOD=	122	123	124	EPISODE
#FN	1239	1161	1199	620
#JAC	50	38	38	105

For this particular problem, a maximum of three iterations seems best. Included here are results using Hindmarsh's EPISODE, which required far fewer function evaluations, but considerably more Jacobian evaluation.

Experiment 4: The Jacobian is evaluated whenever any one of the three conditions described in section 1 is satisfied. It was found by monitoring the program while solving problem 2 that an unsatisfactory convergence rate accounted for about 1/3 of the evaluations, and a sufficiently large change in H accounted for the remainder. Letting H' represent that value of H for which (I-HJ) was last evaluated, (I-HJ) is re-computed when H'/H no longer lies in the interval [1/2, 2]. In this experiment, problem 2 was solved using METHOD=133 and EPS=10⁻⁶. I was assigned the values [3/4, 4/3], [2/3, 3/2], [1/2, 2] and [1/3, 3], and a numerical Jacobian was used so that the number of function evaluations reflects the number of evaluations of both f and J. Results are given at t_i=4.10ⁱ, i=0,1,...,9.

i	[3/4, 4/3]	[2/3, 3/2]	[1/2, 2]	[1/3, 3]
	192	138	189	202

1	336	298	375	339
2	490	525	550	526
3	672	701	738	767
4	820	837	934	968
5	963	968	1056	1168
6	1059	1063	1159	1310
7	1135	1170	1273	1391
8	1196	1225	1324	1456
9	1229	1287	1367	1548

For this problem, the two smaller intervals seem more appropriate.

Experiment 5: To determine the effectiveness of the relaxation factor R in the Newton iteration, problem 2 was solved using METHOD=123, EPS=10⁻⁵, 10⁻⁸ and R=2/(1+H/H') and 1 (no relaxation factor). The results are given at t=4x10⁵. The number of function evaluations (#FN), Jacobian evaluations (#JAC), divergent iterations (#DIVGD) and non-convergent iterations (#CONVGD) were:

	EPS=10 ⁻⁵		EPS=10 ⁻⁸	
	R=2/(1+H/H')	R=1	R=2/(1+H/H')	R=1
#FN	675	832	1975	2243
#JAC	27	33	45	55
#DIVGD	5	3	7	12
#CONVGD	3	13	12	20

For both values of EPS substantially more work was required in the case where no relaxation factor was used.

References

- [1] Burrage, K., Butcher, J.C., Chipman, F., "STRIDE: A stable Runge-Kutta integrator for differential equations", Computational Mathematics Report; University of Auckland (to appear).
- [2] — "An efficient Runge-Kutta implementation", Computational Mathematics Report, University of Auckland (to appear).
- [3] Burrage, K., "A special family of Runge-Kutta methods for solving stiff differential equations", BIT 18, 22-41.
- [4] Butcher, J.C., "On the implementation of implicit Runge-Kutta methods", BIT 16, 237-240.
- [5] — "A transformed implicit Runge-Kutta method", computational mathematics Report No. 13, University of Auckland.
- [6] Krogh, F.T., "On testing a subroutine for the numerical integration of ordinary differential equations" JPL Technical Memorandum No. 217.
- [7] Lapidus, L., Seinfeld, J.H., "Numerical solution of ordinary differential equations", Academic Press, N.Y., 1977.

EXPONENTIAL INTERPOLATIONS AND THE NUMERICAL SOLUTION OF O.D.E (SUMMARY)

ARIEH ISERLES

King's College, Cambridge.

Exponential approximations, mainly the Padé approximations, N-approximations and exponentially fitted approximations, have attracted great interest during recent years as a means for the numerical solution of ordinary differential systems.

The common feature of these approximations is that they fit to the exponential and to its derivatives at the origin. Hence, they are, in fact, Hermitian interpolations and not approximations. The high degree of interpolation at the origin is of great benefit if (in the case of either linear or mildly nonlinear autonomous differential systems) the behaviour of the solution is determined by a single principal eigenvalue of the Jacobian matrix.

From the computational point of view, this is not the case if parabolic or hyperbolic partial differential systems are solved by the method of lines. The solution of the derived linear ordinary differential systems is determined (unless the solution is in an asymptotic stage) by a great number of eigenvalues, in the neighbourhood of the origin. Even if the exact spectrum of the matrix is unknown, it is a useful practical assumption that the eigenvalues are more or less spread uniformly in an interval $[-T, 0]$. Out of these eigenvalues, only a smaller subset, contained in $[-T_0, 0]$, $0 < T_0 \ll T$, influence strongly the numerical solution of the system. Hence, it is of practical interest to investigate the use of exponential approximations which fit to the exponential along an equispaced mesh in $[-T_0, 0]$, instead of high-degree interpolation at the origin.

Such exponential interpolations are derived in this paper.

The first result extends the C-polynomial theory of Norsett to the interpolatory case:

Theorem 1: If $p(x, c) = \sum_{k=0}^n \frac{1}{k!} p_k(c) x^k$, where $c > 0$, $p_n(c) \equiv 1$ and $p_0(c), \dots, p_{n-1}(c)$ are arbitrary functions of c and

$$(1) \quad R(x, c, p) := \frac{\sum_{k=0}^n (-1)^k D_x^{n-k} p(0, c) (1 - e^{-c})^k (-x/c)_k}{\sum_{k=0}^n (-1)^k D_x^{n-k} p(1, c) (e^c - 1)^k (-x/c)_k}$$

where $(-y)_k$ is the factorial function, $(-y)_0 = 1$, $(-y)_k = (-y)(-y+1) \dots (y+k-1)$ for $k \geq 1$ and $D_x^{n-k} p(a, c)$ is the $(n-k)$ -th derivative of $p(x, c)$ at (a, c) , with respect to x , then

$$R(qc, c, p) = e^{-qc}, \quad 0 \leq q \leq n.$$

Definition: The function $p(x, c)$, as defined in Theorem 1, is called the C-function of the exponential interpolation (1). If $p(x, c)$ is continuous for $c > 0$ (or an interval $0 < c < C$) and $\lim_{c \rightarrow 0^+} p(x, c)$ exists, the C-function is said to be smooth.

Theorem 2: If p is a smooth C-function, let $\tilde{p}(x) = \lim_{c \rightarrow 0^+} p(x, c)$ and let $\tilde{R}(x, \tilde{p})$ be the

exponential approximation which is generated by the C-polynomial $\tilde{p}(1-x)$. Then, if for all sufficiently small c , and for a positive integer s ,

$$R(qc, cp) = e^{-qc}, \quad 0 \leq q \leq n + s, \quad \text{then} \\ \tilde{R}(x, \tilde{p}) - e^{-x} = O(x^{n+s+1}).$$

Theorem 3: Every rational function $R(x) = P(x)/Q(x)$, that satisfies the conditions $\deg P_c \leq n$, $\deg Q_c = n$, and $R_c(qc) = e^{-qc}$, $0 \leq q \leq n$, can be expressed in the form (1), and the corresponding C-function is unique. Moreover, if R_c is continuous in the parameter c , then its C-function is smooth.

This theory of C-functions makes it possible to derive explicit formulae for several families of exponential interpolations, corresponding to the known formulae for exponential approximations; two important cases are given in the next two theorems.

Theorem 4: If

$$P_{n,m}(x, c) := \sum_{k=0}^m \frac{(n+m-k)! m!}{(n+m)! k! (m-k)!} (1 - e^{-c})^k \left(-\frac{x}{c}\right)_k,$$

$$Q_{n,m}(x, c) := \sum_{k=0}^n \frac{(n+m-k)! n!}{(n+m)! k! (n-k)!} (1 - e^{-c})^k \left(-\frac{x}{c}\right)_k$$

and $R_{n,m}(x,c) := P_{n,m}(x,c)/Q_{n,m}(x,c)$ then

$$R_{n,m}(qc) = e^{-qc}, \quad 0 \leq q \leq n+m.$$

This rational approximation is analogous to Padé approximation, because all the parameters of $R_{n,m}$ are used to satisfy interpolation conditions. Therefore we call this case "Padé interpolation".

Theorem 5: If $a > 0$,

$$P_n(x,c,a) := \sum_{k=0}^n \frac{1}{k!} \left(\sum_{p=0}^k (-1)^p \binom{k}{p} (1+acp)^n e^{-pc} \right) \left(-\frac{x}{c} \right)_k$$

$$\text{and } R_n(x,c,a) := P_n(x,c,a) / (1+ax)^n$$

$$\text{then } R_n(qc,c,a) = e^{-qc}, \quad 0 \leq q \leq n.$$

Further, if a_0 satisfies

$$\sum_{k=0}^{n+1} (-1)^k \binom{n+1}{k} (1+a_0ck)^n e^{-kc} = 0,$$

$$\text{then } R_n((n+1)c,c,a_0) = e^{-(n+1)c} \text{ ("N-interpolations").}$$

A-acceptability: As yet, no general conditions have been found that imply A-acceptability. Various computer results are available, concerning the range of c which yields A-acceptability and A_0 -acceptability for various Padé and N-interpolations.

Arieh Iserles
King's College
Cambridge CB2 1ST
England.

On the Existence of Maximum Polynomial Degree Nordsieck-Gear

(k,p) Methods for All (k,p)

Roy Danchick and Mario L. Juncosa

The Rand Corporation

Santa Monica, California 90406

Introduction

This paper sketches a fundamental theorem on the existence of Nordsieck-Gear (k,p) methods of maximal polynomial degree. The notation (k,p) refers to the number of scaled derivatives retained by the method (k) and the order of the differential equation (p). The method's polynomial degree is the highest degree polynomial that can be numerically integrated exactly; given exact starting values.

In [1] it was shown, for fixed stepsize and for sufficiently accurate starting values, that if a certain sequence of corrector-predictor coefficients was well-defined, then the Nordsieck-Gear (k,p) methods could obtain a maximum polynomial degree of k+1. The sequence was shown to be well-defined in the (k,1) case for k=2,3,4, and 5. These results led to the conjecture that such a well-defined sequence exists for all (k,p). The results of this paper prove that conjecture.

Basic Definitions

I is the $(k+1) \times (k+1)$ identity matrix

$$\ell^{(n+1)} = [\ell_0^{(n+1)}, \ell_1, \ell_2, \dots, \ell_k]^T,$$

$$e_0 = [1, 0, 0, \dots, 0]^T,$$

$$e_1 = [0, 1, 0, \dots, 0]^T.$$

A_k is the $(k+1) \times (k+1)$ Pascal triangle matrix with element a_{ij} given by

$$a_{ij} = \begin{cases} \binom{j}{i} & \text{if } 0 \leq i \leq j \leq k, \\ 0 & \text{if } 0 \leq j < i \leq k, \end{cases}$$

$$\{ \binom{k+1}{i} \} = [\binom{k+1}{0}, \binom{k+1}{1}, \dots, \binom{k+1}{k}]^T.$$

Preliminary Discussion

The Nordsieck-Gear (k,1) method for the numerical solution of the scalar O.D.E.,

$$\bar{y}'(x) = f(x, \bar{y}(x)), \quad \bar{y}(0) = \bar{y}_0,$$

for fixed stepsize, h, can be written as

$$Y_{n+1}^{(0)} = AY_n, \quad (1)$$

$$Y_{n+1}^{(i+1)} = Y_{n+1}^{(0)} + [hf((n+1)h, e_0^T Y_{n+1}^{(i)}) - e_1^T Y_{n+1}^{(0)}] \ell^{(n+1)}.$$

The stability coefficients $\ell_1, \ell_2, \dots, \ell_k$ are chosen so that the characteristic polynomial of $(I - \ell^{(n+1)} e_1^T) A_k$ is $(\lambda - 1)\lambda^k$. The polynomial degree maximizing coefficient $\ell_0^{(n+1)}$ is computed from the difference equation

$$S_0 = 0 \quad (2)$$

$$S_{n+1} = (I - \ell^{(n+1)} e_1^T) [AS_n - \{ \binom{k+1}{i} \}],$$

according to

$$\ell_0^{(n+1)} = (1 - e_0^T AS_n) / (k+1 - e_1^T AS_n), \quad (3)$$

(2) following from a recursion argument on the error in solving the ODE $\bar{y}'(x) = (k+1)x^k, \bar{y}_0 = 0$.

We now sketch the proof that the choice of the polynomial degree maximizing coefficient $\ell_0^{(n+1)}$ according to (3) can always be made.

Definitions

\bar{I} is the $k \times k$ identity matrix,

$$\bar{\ell} = [\ell_1, \ell_2, \dots, \ell_k]^T,$$

$$\bar{e}_1 = [1, 0, \dots, 0]^T.$$

\bar{A}_k is lower $k \times k$ diagonal submatrix of A_k ,

$$\{ \binom{k+1}{i} \} = [\binom{k+1}{1}, \binom{k+1}{2}, \dots, \binom{k+1}{k}]^T.$$

K is the $k \times k$ Jordan canonical form companion matrix to the polynomial λ^k .

P is the $k \times k$ permutation matrix of columns of the identity matrix written in reverse order

V_{k-1} is the Vandermonde matrix with element

$$v_{ij} = i^j \quad (i=0, 1, 2, \dots, k-1, j=0, 1, 2, \dots, k-1),$$

$$\bar{e} = [1, 1, \dots, 1]^T.$$

Theorem

Maximum Polynomial Degree Nordsieck-Gear
(k,p) Methods exist for all (k,p).

Sketch of Proof

The idea behind the proof is to reduce the lower kxk submatrix of $A_k(I - \ell^{(n+1)} e_1^T)$ to Jordan canonical form leaving $e_1^T A_n$ invariant and to show that this term is non-positive. Since A_k is upper triangular the problem can be reduced by one in dimension by defining the sequence.

$$\bar{S}_0 = 0, \quad (4)$$

$$\bar{S}_{n+1} = (\bar{I} - \bar{\ell} e_1^T) [\bar{A}_n - \{(\bar{k+1}_i)\}].$$

Let $W_n = L_k Q_k \bar{A}_k \bar{S}_n$. Here Q_n is a unique upper triangular matrix with first row equal to \bar{e}_1^T . It takes $\bar{A}_k(\bar{I} - \bar{\ell} e_1^T)$ into the form

$$B_k = Q_k \bar{A}_k (\bar{I} - \bar{\ell} e_1^T) Q_k^{-1} = I + K - \{(\bar{k+1}_i)\} \bar{e}_1^T. \quad (5)$$

B_k is similar under P to the companion matrix of the polynomial λ^k . L_k is a unique lower triangular matrix, whose first row is \bar{e}_1^T , that takes B_k into K according to

$$K = L_k B_k L_k^{-1}. \quad (6)$$

Under $L_k Q_k \bar{A}_k$ the \bar{S}_n sequence goes into

$$W_{n+1} = K[W_n - L_k Q_k \{(\bar{k+1}_i)\}]. \quad (7)$$

Moreover

$$\bar{e}_1^T W_n = \bar{e}_1^T L_k Q_k \bar{A}_k \bar{S}_n = \bar{e}_1^T Q_k \bar{A}_k \bar{S}_n = \bar{e}_1^T \bar{A}_k \bar{S}_n = \bar{e}_1^T A_k S_n. \quad (8)$$

The theorem is proved if it can be shown that $\bar{e}_1^T W_n \leq 0$ for $n = 0, 1, 2, \dots$. A sufficient condition for this sequence of inequalities to hold, given (4), is that

$$L_k Q_k \{(\bar{k+1}_i)\} \geq 0. \quad (9)$$

This is accomplished by showing that

$$L_k Q_k \{(\bar{k+1}_i)\} = (k+1) L_k Q_k L_k^{-1} e \quad (10)$$

and, by induction on k, that

$$L_k Q_k L_k^{-1} \geq 0. \quad (11)$$

The passage to the $k > p > 1$ case is made by showing that the general form of the Q_k associated with a (k,p) method is

$$Q_k = (A_{k-p}^T) V_{k-p} \text{diag}[(\bar{p}_p), (\bar{p+1}_p), \dots, (\bar{k}_p)] \quad (12)$$

and that for the (k,p) case

$$\{(\bar{k+1}_i)\} = (\bar{k+1}_p) \text{diag}[(\bar{p}_p), (\bar{p+1}_p), \dots, (\bar{k}_p)]^{-1} L_{k-p}^{-1} \bar{e}. \quad (13)$$

This reduces the general (k,p) case to the (k-p,1) case.

References

- (1) Danchick, R., "On the Non-Equivalence of Maximum Polynomial Nordsieck-Gear and Classical Method," Proceedings of the Conference on the Numerical Solution of Ordinary Differential Equations 17, 20 October 1972, University of Texas at Austin, Lecture Notes in Mathematics, Springer, Vol. 362, No. VIII, pp. 92-106.

TWO DEVICES FOR IMPROVING THE EFFICIENCY OF STIFF ODE SOLVERS

Peter Alfeld
Department of Mathematics
University of Utah
Salt Lake City, Utah

ABSTRACT

The nonlinear systems of equations that arise when a stiff system of ordinary differential equations is tackled by an implicit linear multistep method is normally solved using a modified version of Newton's method in which the Jacobian is kept constant for as many steps as possible. An initial approximation is obtained by applying a predictor, typically an explicit linear multistep method. In this talk, two approaches to improving the efficiency of the above procedure are described.

Firstly, it is suggested to use quasi-Newton methods that update the Jacobian approximation at each iteration-step for the solution of the nonlinear system. It is argued that the most popular such method, Broyden's first method, is unsuitable for the special nonlinear systems under consideration, and that instead Broyden's second method should be used. That argument depends very critically on the different roles that are played by the eigenvalues (of the Jacobian) with large and small, respectively, absolute values. Computational aspects are considered, and numerical examples are given.

Secondly, it is suggested to improve the initial approximation by interpolating corrections that were applied to previously obtained initial approximations. This device is motivated by considering the test equation $y' = \lambda y$. Numerical examples suggest that it works very well.

1. INTRODUCTION

When an implicit linear multistep method

$$\sum_{j=0}^k \alpha_j y_{n+j} = h \sum_{j=0}^k \beta_j f_{n+j} \quad (1)$$

is applied to a stiff initial value problem

$$y' = f(y); y(a) = \eta \in \mathbb{R}^m; x \in [a, b] \quad (2)$$

then a nonlinear system of equations

$$F_n(y_{n+k}) = y_{n+k} - h \beta_n f(x_{n+k}, y_{n+k}) + \sum_{j=0}^{k-1} (\alpha_j y_{n+j} - h \beta_j f_{n+j}) = 0 \quad (3)$$

has to be solved at each step of the integration (1).

2. THE USE OF QUASI-NEWTON METHODS

In this talk, it is proposed to solve the nonlinear system (3) by a rank-one quasi-Newton method defined by:

- 1) $y_{n+k}^{[0]}$ given by some prediction operator.

Let $r = 0$.

- 2) Solve $B_{n+k}^{[r]} s_{n+k}^{[r]} = -F_n(y_{n+k}^{[r]})$ where $B_{n+k}^{[r]}$

is an approximation to $F'_n(y_{n+k}^{[r]})$.

- 3) Let $y_{n+k}^{[r+1]} = y_{n+k}^{[r]} + s_{n+k}^{[r]}$

$$u_{n+k}^{[r]} = F_n(y_{n+k}^{[r+1]}) - F_n(y_{n+k}^{[r]})$$

(4)

$$B_{n+k}^{[r+1]} = B_{n+k}^{[r]} + \frac{(u_{n+k}^{[r]} - B_{n+k}^{[r]} s_{n+k}^{[r]})(v_{n+k}^{[r]})^T}{(s_{n+k}^{[r]})^T v_{n+k}^{[r]}}$$

(where $v_{n+k}^{[r]}$ is specified below).

- 4) Check for convergence. If not satisfied let $r = r+1$ and go back to step 2).

A detailed discussion of quasi-Newton methods may be found in [1], [3], [4], [6]. Among others, the following properties are important in the present context: 1) Quasi-Newton methods converge superlinearly. 2) The update of the Jacobian can be replaced by an update of the inverse Jacobian.

Normally, a version of Newton's method is used for the numerical solution of (3). The Jacobian is kept constant for as many steps as possible, but when it has to be reevaluated, or reestimated, this requires *additional function evaluations*. In a quasi-Newton method, this expensive procedure is replaced by a cheap update of the Jacobian at each iteration for the solution of the non-linear system (3). This is more efficient as it requires *no additional function evaluations*.

Broyden [3] suggests the following choices of the update (4):

$$v_{n+k}^{[r]} = s_{n+k}^{[r]} \quad (\text{Broyden's first method}) \quad (5)$$

$$v_{n+k}^{[r]} = (B_{n+k}^{[r]})^T u_{n+k}^{[r]} \quad (\text{Broyden's second method}) \quad (6)$$

Broyden's first method is generally considered the better *general purpose method* (see [3], [5]).

However, in [1] it is argued that for the *special nonlinear systems* under consideration Broyden's second method is more suitable.

For lack of space the argument can only be outlined here. (A full description may be found in [1].) It is based on considering the different roles that are played by the eigenvalues of the Jacobian F'_n with large and small, respectively, absolute values. The argument can be summarized in three points:

- 1) A prediction $y_{n+k}^{[0]}$ can be obtained that is particularly accurate in components corresponding to eigenvalues of large absolute value.
- 2) Any quasi-Newton method reduces in a certain sense to the secant method applied to each component (corresponding to one of the eigenvalues) of the solution of (3). For the secant method, the error at the r -th stage is roughly proportional to the previous two errors and inversely proportional to the first derivative of the function under investigation. In the present context, this means that the error in the component corresponding to an eigenvalue λ_i , say, is roughly proportional to the two previous errors in that component and inversely proportional to λ_i .

The above points show that the eigenvalues of F'_n with large absolute value have little importance for the convergence of the iteration for the solution of (3). (They do matter for stability). The key point is the following:

- 3) Broyden's second method approximates particularly well the eigenvalues of small absolute value. More precisely: Broyden's first method minimizes (over all rank-one updates) a sharp bound on the error of the eigenvalues, whereas Broyden's second method minimizes a similar bound on the *reciprocals* of the eigenvalues. Since these bounds are independent of the eigenvalues, Broyden's first method yields small relative errors of the large eigenvalues, and Broyden's second method yields small relative errors of the small eigenvalues (large and small in absolute value).

In [1], numerical examples (taken from [7]) are given that confirm that quasi-Newton methods are strong competitors of Newton's method, and that indeed the update (6) performs better than the update (5) (in the present special context).

Numerical aspects considered in [1] include the following:

- 1) Rather than updating $B_{n+k}^{[r]}$ in (4), we can update $(B_{n+k}^{[r]})^{-1}$, thus saving an LU-factorization at each step.
- 2) When the step-size or order of (1) is changed, then it is a viable strategy not to change the Jacobian approximation at all. The basis for this device is the fact that the smaller the absolute value of an eigenvalue, the less it is affected by such a change.

3. THE USE OF INTERPOLATION

In [2], the following modification of the normal prediction process is described:

$$\tilde{y}_{n+k} \quad \text{given by an explicit linear multistep method} \quad (7(i))$$

$$y_{n+k}^{[0]} = \tilde{y}_{n+k} + \tilde{d}_{n+k} \quad (7(ii))$$

$$y_{n+k} \quad \text{obtained by applying a variant of Newton's method or a quasi-Newton method to (3).} \quad (7(iii))$$

\tilde{d}_{n+k} is obtained by interpolating

$d_{n+j} = y_{n+j} - \tilde{y}_{n+j} \quad (j < k)$. This approach depends on d_{n+j} being a smooth function of x_{n+j} . Using an asymptotic expansion of the

global truncation error for investigating this question fails for stiff systems.

Therefore, the test equation

$$y' = \lambda y \quad (8)$$

is considered. It turns out that the corrections d_{n+k} show the same qualitative behaviour as the numerical solution of (8) by (1).

It follows that, for example, Gear's backward differentiation formulas (see [8]) are suitable for the approach (7). On the other hand, the Trapezoidal Rule leads to oscillating global errors, and thus to oscillating corrections d_{n+j} . However, these oscillations occur only for eigenvalues of large absolute value. Since we have seen in section 2. that only eigenvalues of small absolute value matter for convergence, it follows that the device (7) can also be applied to the trapezoidal rule. Moreover, particular predictors can be designed that dampen the oscillations in the corrections.

In [2] numerical examples are given that confirm the above statements, and the viability of the approach (7).

REFERENCES

- [1] ALFELD, P., *Quasi-Newton Methods for Stiff Ordinary Initial Value Problems*, submitted for publication.
- [2] ALFELD, P., *A Device for Improving the Efficiency of Stiff O.D.E. Solvers*, in preparation.
- [3] BROYDEN, C. G., *A class of methods for solving nonlinear simultaneous equations*, Math. Comp. 19 (1965), pp. 577-593.
- [4] BROYDEN, C. G., *Quasi-Newton Methods*, in W. Murray (ed.), "Numerical Methods for Unconstrained Optimization", Academic Press, London and New York, 1972, pp. 87-106.
- [5] DENNIS, J. E., *A brief introduction to quasi-Newton methods*, in Proceedings of Symposia in Applied Mathematics, V. 22, American Mathematical Society, Providence, Rhode Island, 1978, pp. 19-52.
- [6] DENNIS, J. E., and J. J. MORE, *Quasi-Newton Methods, Motivation and Theory*, SIAM Review 19 (1977), pp. 46-89.
- [7] ENRIGHT, W. H., T. E. HULL, and B. LINDBERG, *Comparing numerical methods for stiff systems of ODEs*, BIT 19 (1976), pp. 10-48.
- [8] GEAR, G. W., *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

Shohei FUJITA

Department of Computer Science, Faculty of Engineering
Tokyo Institute of Technology
O-okayama 2-12-1, Meguro-ku, Tokyo 152, JAPAN

Abstract — Several computational methods based on the backward differentiation formulas have been proposed for the numerical solution of stiff systems of ordinary differential equations. These methods, however, suffer from serious difficulty in that they are unable to cope with the stiff systems in the presence of the eigenvalues of the Jacobian which are close to the imaginary axis. This paper explicates this phenomenon from a point of view of structural stability. A necessary and sufficient condition is presented for structural stability for n -dimensional system of linear ordinary differential equations which is used as a test problem. This proves that the region of stiff stability contains structurally unstable systems. The poor performance of the algorithms when some of the eigenvalues of the Jacobian are close to the imaginary axis is caused by the presence of structurally unstable systems. Furthermore, a reliability of testing for stiff systems is discussed from a point of view of topological equivalence. A practical problem arising from dynamic security assessment for the large-scale interconnected power system is also discussed.

1. Introduction

The purpose of this paper is to demonstrate importance and usefulness of the qualitative theory of differential equations, i.e., the theory of structural stability, in the design of numerical algorithms for stiff differential systems.

2. Structural Stability for n -dim. LODEs

After the introduction of the concept of structural stability by Andronov and Pontryagin, many studies have been devoted to the problem of structural stability for dynamical systems in the field of pure mathematics (e.g., [12]). It seems, however, that there is no description of a necessary and sufficient condition for n -dimensional linear ordinary differential equations (LODEs) to be structurally stable. The purpose of this section is to investigate the concept of structural stability for n -dim. LODEs. First, the concepts and definitions pertaining to the structural stability, which are useful in the following discussions, are summarized briefly. Secondly, we give a necessary and sufficient condition for the structural stability

for n -dim. LODEs and present a clear proof of this result based on the topological equivalence of linear flows [7].

We shall begin by explaining a general concept of "structural stability". A flow ϕ on a topological space M is a continuous map:

$$\phi: R \times M \rightarrow M$$

such that

$$(i) \phi(0, x) = x$$

$$(ii) \phi(t_1, \phi(t_2, x)) = \phi(t_1 + t_2, x)$$

for $x \in M$ and $t_1, t_2 \in R$, where R is the real line. A trajectory of ϕ through $x \in M$ is a canonically ordered set $\{\phi(t, x) \mid t \in R\}$.

Definition 1: Two flows ϕ and ψ on M are said to be topologically equivalent if there exists a homeomorphism of M onto itself carrying each trajectory of ϕ to a trajectory of ψ together with preserving orientation.

For linear flows Definition 1 is shown to be equivalent to the following definition:

Definition 1': Two flows ϕ and ψ on M are said to be topologically c -equivalent for $c > 0$ if there exists a homeomorphism:

$$h: M \rightarrow M$$

such that

$$h \cdot \phi(ct, x) = \psi(t, h(x)), \quad x \in M, t \in R.$$

Let $V(M)$ be the set of all C^1 -vector fields on M . A C^1 -vector field $f \in V(M)$ generates a flow.

Definition 2: A nbhd of $f \in V(M)$ is any subset $N(f)$ in $V(M)$ which contains a set of the form: $\{g \in V(M) \mid \|g - f\| < \delta\}$ for $\delta > 0$, where $\|\cdot\|$ is some norm.

Definition 3: A vector field $f \in V(M)$ is structurally stable if there exists a nbhd $N(f)$ of f in $V(M)$ such that for every $g \in N(f)$, the flows of f and g are topologically equivalent.

Now we can show a new result pertaining to n -dim. LODEs. Let us consider a family of n -dim. LODEs:

$$\dot{x} = Fx, \quad x \in R^n, \quad F \in L(R^n) \quad (1)$$

which is used as a test problem, where $L(R^n)$ denotes

* This work was supported in part by the Ministry of Education, Japan, under Grant 239002.

the set of $n \times n$ matrices. Then we have the following theorem:

Theorem 1: The n -dim. LOEs (1) is structurally stable if and only if

$$\operatorname{Re} \lambda_i(F) \neq 0, \quad i = 1, \dots, n,$$

where $\operatorname{Re} \lambda_i(\cdot)$ denotes the real part of the eigenvalues of the matrix.

Remark 1: It seems that there is no description of a necessary and sufficient condition for the n -dim. LOEs (1) to be structurally stable. The structural stability for n -dim. OEs was investigated in [11] which however depends on the "ε - homeomorphism".

Proof of Theorem 1: First, we note that the vector field of the n -dim. LOEs (1) is $f(x) = Fx$, the flow is specified by $\phi(t, x) = e^{Ft}x$, and $V(M)$ is the set of all $n \times n$ matrices, which we denote by $L(R^n)$. The nbhd of $F \in L(R^n)$ is any subset $N(F)$ in $L(R^n)$ which contains a set of the form: $\{G \in L(R^n) \mid \|F - G\| < \delta\}$ for some $\delta > 0$ and $\|\cdot\|$ uniform norm.

Lemma 1 [7]: Let ϕ and ψ be flows on R^n generated by $\dot{x} = Fx$, $x \in R^n$, and $\dot{y} = Gy$, $y \in R^n$, respectively. Then, the two flows ϕ and ψ are topologically equivalent if and only if $\operatorname{In}(F) = \operatorname{In}(G)$ and $G^* = cF^*$, where $\operatorname{In}(\cdot)$ is the inertia of the matrix, c is a positive constant, and F^* denotes the real Jordan form of the matrix F corresponding to the eigenvalue with zero real part.

We are now in a position to prove Theorem 1:
(Necessity) Assume that the n -dim. LOEs (1) is structurally stable but $\operatorname{Re} \lambda_i(F) = 0$ for some i . By Definition 3 it follows that there exists a nbhd $N(F)$ of $F \in L(R^n)$ such that for every $G \in N(F)$ the flows of F and G are topologically equivalent. By choosing any nbhd $N(F)$ of F , however, there is an $G \in N(F)$ which does not satisfy the condition in Lemma 1. This contradicts to Lemma 1. Thus we conclude that $\operatorname{Re} \lambda_i(F) \neq 0$, $i = 1, \dots, n$, for the structurally stable n -dim. LOEs (1).
(Sufficiency) Suppose $\operatorname{Re} \lambda_i(F) \neq 0$, $i = 1, \dots, n$. Then there exists a nbhd $N(F)$ in $L(R^n)$ such that for any $G \in N(F)$, $\operatorname{Re} \lambda_i(G) \neq 0$, $i = 1, \dots, n$, and $\operatorname{In}(F) = \operatorname{In}(G)$. Hence, by Lemma 1 it follows that the flows of F and G are topologically equivalent.

3. Stiff Stability Contains Structurally Unstable Systems

The most popular methods for solving stiff differential systems are based on the backward differentiation formulas. The crucial drawback of these methods is the poor stability property when the Jacobian matrix has eigenvalues close to the imaginary axis (e.g., [1], [2], [9], [13]). A reason of this difficulty can be interpreted as follows.

First, in view of Theorem 1 and the definition of stiff stability ([5], [8]), we have the following theorem:

Theorem 2: The region of stiff stability contains structurally unstable systems.

Example 1: Consider the 4-dim. LOEs:

$$\dot{x} = \Lambda \Lambda^{-1} x, \quad x \in R^4, \quad t \in [0, 20/\alpha] \quad (2)$$

$$\Lambda = \operatorname{diag}[1, -1, -\alpha + i\omega, -\alpha - i\omega], \quad \alpha > 0$$

$$A = \begin{bmatrix} -1 & -1 & i & -i \\ 1 & 1 & i & -i \\ i & -i & 1 & 1 \\ 1 & -i & -1 & -1 \end{bmatrix}, \quad x(0) = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

The eigenvalues are

$$\lambda_{1,2} = \pm i, \quad \lambda_{3,4} = -\alpha \pm i\omega$$

Hence, by Theorem 1, the LOEs (2) is structurally unstable. We can find, in fact, tests examples where the BDF of order 4 to 6 for the 4-dim. LOEs (2) come to unstable [14].

To summarize: the implicit BDF of order 1 to 6 are stiffly stable, so that by Theorem 2, structural instability is possible for the BDF of order 1 to 6.

4. Testing - Topological Equivalence

An ultimate aim in discussing stability of numerical methods is to determine the stability of the methods for nonlinear systems:

$$\dot{x} = f(x), \quad x \in R^n, \quad x(0) = x_0 \quad (3)$$

The property we would like to demand from a method is that, for a given step-length and order, the global error remains as small as possible. In order to forecast the behavior of a method on a nonlinear system, it is customary to consider the test equation:

$$\dot{x} = \lambda x, \quad x \in R^1 \quad (4)$$

for all λ in some set including the set of eigenvalues of the Jacobian matrix $J = \partial f / \partial x$. This test problem is, however, too simple to give a reliable indication of the performance for stiff systems unless some additional conditions are satisfied. Little has been rigorously proved concerning error propagation in the numerical solution of nonlinear stiff systems [1]. There does not exist a general theory of stability of numerical methods such that the stability behavior of the method when applied to nonlinear stiff systems can be determined in a systematic manner. What are conditions under which the (local) behavior of the nonlinear stiff problem (3) is determined by the solution of the linearized system? This question is partially answered by the topological equivalence Lemma:

Lemma 2 [6]: The local behavior of the NODE (3) near the singular points is topologically equivalent to the behavior of the linearization of (3) if and only if $\operatorname{Re} \lambda_i(J) \neq 0$, $i = 1, \dots, n$.

Moreover, let π_+ , π_0 , and π_- denotes the number of the eigenvalues of the matrix J with positive, zero, and negative real parts, respectively. The topological type of singular points is determined by the integers π_+ , π_- , and by the behavior of the phase trajectories on some invariant submanifold whose dimension is equal to π_0 . When the linearized system has an eigenvalue with real part zero, knowing about the behavior of the linearization of (3) at singular points gives no information about the behavior of (3) itself near the singular points.

Example 2: Let us consider a practical problem arising from dynamic security assessment for the large-scale interconnected power system. A mathematical model for the power system is described by

$$\begin{aligned} M_i \ddot{\delta}_i &= P_{mi} - E_i^2 G_{ii} - D_i \dot{\delta}_i \\ &- \sum_{\substack{j=1 \\ j \neq i}}^n A_{ij} \cos(\delta_i - \delta_j - \phi_{ij}) \end{aligned} \quad (5)$$

$i = 1, \dots, n.$

One of the problems of dynamic security assessment is to estimate critical clearing times. Hence, we must know the behavior of the solution of the second-order NODEs (5) which is possibly unstable due to contingency. If the effects of damping torque is approximated by "uniform damping", i.e.,

$$D_i/M_i = C, \quad i = 1, \dots, n,$$

then the NODEs (5) result in

$$\ddot{x} = f(x) - C\dot{x}, \quad x \in R^{n-1}. \quad (6)$$

The local behavior of the NODEs (6) near singular points is topologically equivalent to the behavior of the linearization of (6) if and only if

$$\begin{aligned} \operatorname{Re} \lambda_i(K_0) &\neq 0, \quad i = 1, \dots, 2n-2, \\ K_0 &= \begin{bmatrix} 0 & I \\ J_0 & -CI \end{bmatrix}. \end{aligned}$$

On the other hand, when all eigenvalues of J_0 are real, to each negative eigenvalue of J_0 there corresponds to either two negative real eigenvalues of K_0 or to a pair of complex conjugate eigenvalue with real part equal to $-C/2$. Thus, in the case of without damping, i.e., $C = 0$, it is possible that

$$\operatorname{Re} \lambda_i(K_0) = 0.$$

Hence, we need a full nonlinear system to obtain information about the behavior of (6) itself near the singular points.

5. Concluding Remarks

The qualitative theory of differential equations, i.e., the theory of structural stability is an important factor which must not be over looked in designing algorithms for stiff differential systems. It may be hoped that the theory of structural stability makes it possible to give a theoretical foundation to the idea of P.E. Zadunaisky in estimating global errors; and may be also useful to study the dangerous property ([1], [10]) of numerical methods for stiff differential systems with eigenvalues which change from large negative values to large positive values during computation.

References*

- [1] Dahlquist, G. Problems related to the numerical treatment of stiff differential equations. International Computing Symposium 1973, (Günter, A. et al. eds.), North-Holland Publ. Co., 1974

- [2] Enright, W. H., Hull, T. E. and Lindberg, B. Comparing numerical methods for stiff systems of O.D.E.s. BIT, vol.15, pp.10 - 48, 1975.
- [3] Fujita, S. Structural stability and genericity for n-dim. dynamical systems, with applications to system theory. in Research Reports for Coordinate Research Program (A) (supported by the Ministry of Education, Japan, under Grant 239002), pp.117 - 144, 1978.
- [4] Fujita, S., Tsuji, K. and Fukao, T. Topological equivalence of state spaces in the interconnected power system in emergency control. IEEE Trans. Automat. Contr., vol. AC-24, no.3, 1979 (to appear).
- [5] Gear, C. W. Numerical Initial Value Problems in Ordinary Differential Equations. Prentice-Hall, 1971.
- [6] Hartman, P. Ordinary Differential Equations. John Wiley & Sons, 1964 & 1973.
- [7] Ladis, N. N. The topological equivalence of linear flows. Diff. Urav., vol.9, pp.1222 - 1235, 1973.
- [8] Lambert, J. D. Computational Methods in Ordinary Differential Equations. John Wiley & Sons, 1973.
- [9] ———. The initial value problem for ordinary differential equations. in The State of the Art in Numerical Analysis (Jacobs, D. A. H. ed.), Academic Press, 1977.
- [10] Lindberg, B. On a dangerous property of methods for stiff differential equations. BIT, vol.14, pp.430 - 436, 1974.
- [11] Markus, L. Structurally stable differential systems. Ann. Math., vol.73, pp.1 - 19, 1961.
- [12] Peixoto, M. M. (ed.). Dynamical Systems. Academic Press, 1973.
- [13] Skeel, R. D. and Kong, A. K. Blended linear multistep methods. ACM Trans. Math. Soft. vol.3, no.4, pp.326 - 345, 1977.
- [14] Skelboe, S. The control of order and step-length for backward differentiation methods. BIT, vol.17, pp.91 - 107, 1977.

* An extensive bibliography up to 1973 can be found in Stiff Differential Systems (Willoughby, R. A. ed.), Plenum Press, 1974.

On the Theory and the Construction
of Cyclic Multistep Methods

Peter Albrecht
Nuclear Research Center Jülich

A general theory of high order cyclic methods is given in /1/ and /2/; in this contribution we want to present some practical consequences. The theory is based upon the result that the functional ("A-norm")

$$\hat{A}[\delta] := \max_{0 \leq j \leq p} w_j; \quad w_j := \hat{A}^j \delta_0 + h \sum_{l=1}^j \hat{A}^{j-l} \delta_l; \quad \delta_j \in \mathbb{R}^s \quad (1)$$

which depends upon a real (s,s)-matrix \hat{A} with $\|\hat{A}\| = \text{const.}$, $j \in \mathbb{N}$, is (the most adequate) stability functional for a very wide class of discretization methods, namely the \hat{A} -dependent methods ("A-methods") defined by

$$z_0 = z_0(h); \quad z_j = \hat{A} z_{j-1} + h \Phi(x_{j-1}, z_{j-1}, z_j; h), \quad j=1(1)p \quad (2a)$$

where $z_0(h), z_j \in \mathbb{R}^s$, $h \in (0, h_0] \subseteq (0, b-a]$ and $\Phi: [a, b] \times \mathbb{R}^s \times \mathbb{R}^s \times [0, h_0] \rightarrow \mathbb{R}^s$ is continuous and satisfies a Lipschitz-condition w.r.t. the 2nd and 3rd argument.

Frequently it is convenient to write (2a) in the equivalent form:

$$z_0 = z_0(h); \quad L z_j = U z_{j-1} + h \Phi(x_{j-1}, z_{j-1}, z_j; h) \quad (2b)$$

with $\det L \neq 0$. Almost all commonly used methods can be reduced to these matrix forms. As an example consider the following linear 2-cyclic 2-step method:

$$y_{2j+1} + \alpha_1^1 y_{2j-1} + \alpha_0^1 y_{2j-2} = h(\beta_{2f}^1 y_{2j} + \beta_{1f}^1 y_{2j-1} + \beta_{0f}^1 y_{2j-2})$$

$$y_{2j+1} + \alpha_1^2 y_{2j} + \alpha_0^2 y_{2j-1} = h(\beta_{2f}^2 y_{2j+1} + \beta_{1f}^2 y_{2j} + \beta_{0f}^2 y_{2j-1})$$

$$y_0 = y_0, \quad y_1 = y_1(h) \quad j=1(1)p.$$

Combining the linear terms with y_{2j} and y_{2j+1} on the left side and all other terms on the right side, we obtain form (2b) with

$$z_j := \begin{pmatrix} y_{2j} \\ y_{2j+1} \end{pmatrix}; \quad L := \begin{pmatrix} 1 & 0 \\ \alpha_1^2 & 1 \end{pmatrix}; \quad U := \begin{pmatrix} -\alpha_0^1 & -\alpha_0^2 \\ 0 & -\alpha_0^1 \end{pmatrix}$$

$$\Phi_j := \begin{pmatrix} \beta_{2f}^1 y_{2j} + \beta_{1f}^1 y_{2j-1} + \beta_{0f}^1 y_{2j-2} \\ \beta_{2f}^2 y_{2j+1} + \beta_{1f}^2 y_{2j} + \beta_{0f}^2 y_{2j-1} \end{pmatrix}.$$

Multiplication with L^{-1} yields form (2a).

Similarly, any linear M-cyclic k-step methods can be reduced to the forms (2a/b) defining $s := \max(M, k)$ and

$$z_0(h) := (0, \dots, 0, \eta_0, \eta_1(h), \dots, \eta_{k-1}(h))^T \in \mathbb{R}^s \quad (3)$$

$$z_j := (y_{Mj+k-s}, y_{Mj+k-s+1}, \dots, y_{Mj+k-1})^T \in \mathbb{R}^s$$

In the case $M \neq k$ we have to add $(k-M)$ "trivial steps" of the form $y_{Mj+l} = y_{Mj+l}$, $l=0(1)k-M-1$, before reducing to form (2b); (in the special case $M=1$ one thus obtains the well-known one-step representation of a linear k-step method, where \hat{A} is a Frobenius matrix).

An analysis of the functional (1) yields the following theorem which is related to results of Stetter /7/, pg.314, and Skeel /5/:

Theorem: (a) Let $\delta_j \in \mathbb{R}^s$ for all $h \in (0, h_0]$ have the form

$$\delta_0 = \mathcal{O}(h^{q+1}); \quad \delta_j = \mathcal{O}(h^{q+1}) (x_j) + \mathcal{O}(h^{q+1}), \quad j=1(1)p \quad (4)$$

with $q \in \mathbb{N}$, $t \in \mathbb{R}^s$ independent of h and j , and $y^{(q+1)}$ bounded in $[a, b]$.

(b) Let the eigenvalues μ_i of \hat{A} satisfy

$$\mu_1 = 1, \quad |\mu_i| < 1, \quad i=2(1)s. \quad (5)$$

Then $\psi_h^{\hat{A}}[\delta] = \mathcal{O}(h^{q+1})$ if and only if one of the following conditions is satisfied:

$$\text{rank}(I - \hat{A}, t) = s-1 \quad (6a)$$

$$\left| \sum_{l=1}^j y^{(q+1)}(x_l) \right| \leq C, \quad C \text{ indep. of } h \text{ and } j. \quad (6b)$$

Assumption (5) can be weakened to the case of multiple eigenvalues at $\mu=1$, and still other generalizations are possible (see /2/).

As $\psi_h^{\hat{A}}$ is stability functional for all \hat{A} -methods with local discretization errors δ_j , the theorem states, in particular, under which conditions a composite method with stages of order q converges with order $(q+1)$. The decisive conditions (6a) and (6b) will be called "order conditions"; especially condition (6a) turns out to be basic for the construction of high order composite methods.

In what follows, we shall use (6a) for the construction of stiffly stable, cyclic methods with maximal order. Before doing so, however, let us apply our result to certain block-implicit methods in order to give an idea of the usefulness and the wide scope of application of the order condition.

Consider the methods

$$\begin{pmatrix} y_{Mj-M+1} \\ y_{Mj-M+2} \\ \vdots \\ y_{Mj} \end{pmatrix} = \begin{pmatrix} 0 & \dots & 0 & 1 \\ 0 & \dots & 0 & 1 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{Mj-2M+1} \\ y_{Mj-2M+2} \\ \vdots \\ y_{Mj-M} \end{pmatrix} + h \begin{pmatrix} \beta_0^1 f_{Mj-M} + \dots + \beta_M^1 f_{Mj} \\ \beta_0^2 f_{Mj-M} + \dots + \beta_M^2 f_{Mj} \\ \dots & \dots & \dots & \dots \\ \beta_0^M f_{Mj-M} + \dots + \beta_M^M f_{Mj} \end{pmatrix}$$

$$z_j = \hat{A} z_{j-1} + h \hat{\Phi}(x_{j-1}, z_{j-1}, z_j; h)$$

with $j=1(1)p$, $z_0 = (0, \dots, 0, \eta_0)^T \in \mathbb{R}^M$.

These one-step methods have been previously studied by Watts and Shampine /9/ and by Bickart, Burgess and Sloate /3/, /6/.

Let us assume, that all stages have order q except the last one which is supposed to have order $(q+1)$; then $t = (c_{q+1}^1; \dots; c_{q+1}^{M-1}; 0)^T$ ($c_{q+1}^{(r)}$: error const. of the r^{th} stage) and the method converges with order $(q+1)$ because (6a) is satisfied.

Another application of the order condition (6a) is the following: If we construct a k -cyclic k -step method with (unstable) stages of order $(2k-1)$, determining the k free parameters such that the method is stable and satisfies (6a), then we obtain Donelson and Hansen's /4/ k -cyclic k -step methods of maximum order $2k$. Thus, our theorem provides a rather simple and straightforward approach to them, discarding the concept of "auxiliary systems" introduced in /4/.

As the stability regions of these methods are fairly small for $k > 3$, we may ask, whether methods with slightly reduced order possibly have better stability properties. This is the case and, in fact, one can even achieve stiff stability, maintaining unusual high orders. In particular, we have found

- 2-cyclic 2-step methods of order 3
 - 2-cyclic 3-step methods of order 4
 - 3-cyclic 3-step methods of order 5
 - 4-cyclic 4-step methods of order 6.
- (7)

The existence of such methods is remarkable because, up to now, no other stiffly stable, linear methods with order greater than their stepnumber k were known.

We now consider M -cyclic k -step methods of the form

$$z_0 = \zeta(h); \quad Lz_j = Uz_{j-1} + h(B_L F_j + B_U F_{j-1}), \quad j=1(1)p$$

with $F_0 = (0, \dots, 0, f_0, \dots, f_{k-1})^T \in \mathbb{R}^s$; $s = \max(M, k)$;

$$F_j = (f_{Mj+k-s}, \dots, f_{Mj+k-1})^T \in \mathbb{R}^s$$

and (s, s) -matrices L, U, B_L, B_U , where B_L and B_U have a similar structure as L and U and contain the β_1^r .

Tendler, Bickart and Picel /8/ have constructed stiffly stable methods of this type with order k for the special case $B_U = 0$. In the general case one obtains the maximum order methods mentioned in (7); they were constructed in the following way:

Let be μ_i ($i=1(1)s$) the zeros of $P(\mu) := \det(\mu L - U)$ and $\tilde{\mu}_i$ ($i=1(1)s$) those of $\tilde{P}(\mu) := \lim_{H \rightarrow \infty} H^{-M} Q(\mu; H)$, where $H = \lambda h$ and Q is the characteristic polynomial

$$Q(\mu; H) := \det(\mu(L - HB_L) - (U + HB_U)).$$

The order q of the stages is chosen as high as possible and such that a sufficient number of free parameters is available to satisfy the following conditions:

1. $|\mu_i| < 1$, $i=2(1)s$ ("stability at $H=0$ ")[†] (8a)
2. $|\tilde{\mu}_i| < 1$, $i=1(1)s$ ("stability at $H=\infty$ ") (8b)

and, if possible,

3. $\text{rank}(L - U, c) = s-1$ (order condition) (8c)
- $c := (c_{q+1}^1, \dots, c_{q+1}^M)^T$; c_{q+1}^r : error const. of r^{th} stage

(8c) is obtained from (6a) by multiplication with L using $Lt=c$.

Each method that satisfies (8a-c) is then tested on stiff stability performing the mapping of the unit circle into the H -plane by $Q(\mu; H)=0$.

Up to now, a general existence theory of stiffly stable, cyclic methods with maximum order doesn't exist. Also the question of practical performance and implementation have not yet been sufficiently investigated. A series of tests, run at the Nuclear Research Center in Jülich, Germany, furnished promising results.

[†] From consistency we have always $\mu_1 = 1$.

References

- /1/ Albrecht, P.: Explicit, optimal stability functionals and their application to cyclic discretization methods, Computing 19 (1978).
- /2/ Albrecht, P.: Die numerische Behandlung gewöhnlicher Differentialgleichungen, Hauser, München 1979.
- /3/ Bickart, T.A.; Burgess, D.A.; Sloate, H.M.: High order A-stable composite multistep methods for numerical integration of stiff differential equations. Proc. 9th Annual Allerton Conf. on Circuit and System Theory, Univ. of Illinois, 465-473 (1971).
- /4/ Donelson, J.; Hansen, E.: Cyclic composite multistep predictor-corrector-methods. SIAM J. Numer. Anal. 8, 137-157 (1971).
- /5/ Skeel, R.: Analysis of fixed-stepsize methods. SIAM J. Numer. Anal. 13, 271-300 (1976).
- /6/ Sloate, H.M.; Bickart, T.A.: A-stable composite multistep methods. J. Assoc. Comp. Mach. 20, 7-26 (1973).
- /7/ Stetter, H.J.: Analysis of discretization methods for ordinary differential equations. Berlin, Springer 1973.
- /8/ Tendler, J.M.; Bickart, T.A.; Picel, Z.: A stiffly stable integration process using cyclic composite methods. Assoc. Comp. Mach. TOMS Vol. 4 No.4, 339-368 (1978).
- /9/ Watts, H.A.; Shampine, L.F.: A-stable block-implicite one-step methods. BIT 12, 252-266, (1972).

by

ZAHARI ZLATEV

The Royal Veterinary and Agricultural University, Institute of Mathematics and Statistics
Thorvaldsensvej 40, DK - 1871 Copenhagen V, Denmark
and

PER GROVE THOMSEN

Technical University of Denmark, Institute for Numerical Analysis, DK - 2800 Lyngby, Denmark

Consider the non-stiff system of ordinary differential equations

$$(1) \quad y' = f(x, y); \quad y(a) = \eta; \quad x \in [a, b]; \quad f \in S; \quad s \geq 1.$$

Define the grid $a = x_0 < x_1 < x_2 < \dots < x_N = b$ and the stepsize $h_i = x_i - x_{i-1}$, $i = 1(1)N$. The computational work per step (i.e. the work needed to compute an approximation y_i to the true solution $y(x_i)$ of (1) when all y_j , $j = 0(1)i-1$, are already found) is smaller when a constant stepsize ($h_i = h = (b-a)/N$, $i = 1(1)N$) is used. When variable step-sizes are used we accept much more work per step (needed to select an optimal stepsize and to change the stepsize) hoping to minimize the number of steps (i.e. the number N of points in the grid) according to the prescribed error tolerance ϵ . Different formulae can be used in an attempt to optimize the stepsize selection further (this is connected with extra computational work per step too). Such methods are called variable stepsize variable formula methods (VSVFM's). The only linear multistep VSVFM's used in practice are based on the Adams formulae which have many excellent properties: (i) are zero-stable for any change of the stepsize and the formula, see Gear and Watanabe [2]; (ii) an error estimation can easily be obtained, see Gear [1], Krogh [4], Shampine and Gordon [6]; (iii) the storage requirements and the computational work per step are smaller than those of the other linear multistep methods. However, if the absolute stability requirements during the numerical integration of the particular problem are dominant over the accuracy requirements (see Shampine [5]) then the possibility of using large stepsizes will be restricted,

because these methods have small absolute stability regions. An attempt to improve their performance in such situations has been done in [4] and [6] where $P_{kEC_{k+1}E}$ schemes (which have better stability properties, see the plots in [6]) are adopted instead of P_{kEC_kE} schemes. Some linear multistep formulae with large absolute stability regions have been derived by Thomsen and Zlatev [7]. A code based on these formulae is discussed in this paper.

Consider the following $(\alpha_k, \alpha_k^*) - P_{kEC_{k+1}E}$ scheme:

$$(2) \quad y_{n+k}^* = \alpha_k^* y_{n+k-1}^* + (1 - \alpha_k^*) y_{n+k-2}^* + h \sum_{i=0}^{k-1} \beta_{ik}^* f_{n+i};$$

$$(3) \quad f_{n+k}^* = f(x_{n+k}, y_{n+k}^*);$$

$$(4) \quad y_{n+k} = \alpha_k y_{n+k-1} + (1 - \alpha_k) y_{n+k-2} + h \sum_{i=0}^{k-1} \beta_{ik} f_{n+i}$$

$$+ h \beta_{kk}^* f_{n+k}^*;$$

$$(5) \quad f_{n+k} = f(x_{n+k}, y_{n+k}).$$

If $\alpha_k = \alpha_k^* = 1$ then (2) - (5) represent an Adams $P_{kEC_{k+1}E}$ scheme. If α_k and α_k^* ($k = 3(1)12$) are as in Table 1 then the absolute stability regions for the $(\alpha_k, \alpha_k^*) - P_{kEC_{k+1}E}$ schemes are larger than those for the corresponding Adams - $P_{kEC_{k+1}E}$ schemes. A similar (but slightly more complicated than (2) - (5)) scheme can be found for $k=2$ (see [7]). An example is given in Fig. 1 where $k=5$ and the stability formula for our method is plotted by a bold line, the region for the Adams- P_5EC_6E scheme is plotted by a dotted line and the region for the Adams- P_5EC_5E scheme is plotted by dashes; more details can be found in [7], where

the plots for all k ($k=2(1)12$) are given

k	α_k^*	α_k
3	1.36	1.40
4	1.87	1.90
5	1.90	1.95
6	0.55	1.50
7	1.70	1.90
8	1.50	1.90
9	0.50	1.80
10	-2.50	1.50
11	-0.60	1.80
12	-1.00	1.80

Table 1

In the code (2) and (4) are expressed in terms of divided differences. Define ($n+k \leq N$, $i \geq 1$)

$$(6.a) \quad g_{n+k,0} = f_{n+k}; \quad g_{n+k,0}^* = f_{n+k}^*$$

$$(6.b) \quad g_{n+k,i} = (g_{n+k,i-1} - g_{n+k-1,i-1}) / (x_{n+k} - x_{n+k-i});$$

$$(7.a) \quad \Gamma_{0,1} = x_{n+k} - x_{n+k-1}, \quad \tilde{\Gamma}_{0,1} = x_{n+k} - x_{n+k-2};$$

$$(7.b) \quad \Gamma_{0,j} = -\Gamma_{0,1} \Gamma_{0,j-1} (j-1) / j, \quad j=2(1)k+1;$$

$$(7.c) \quad \tilde{\Gamma}_{0,j} = -\tilde{\Gamma}_{0,1} \tilde{\Gamma}_{0,j-1} (j-1) / j, \quad j=2(1)k+1;$$

$$(7.d) \quad \Gamma_{i,j} = (x_{n+k} - x_{n+k-i}) \Gamma_{i-1,j} + \Gamma_{i-1,j+1}, \quad i=1(1)k-1;$$

$$(7.e) \quad \tilde{\Gamma}_{i,j} = (x_{n+k} - x_{n+k-i}) \tilde{\Gamma}_{i-1,j} + \tilde{\Gamma}_{i-1,j+1}, \quad i=1(1)k-1;$$

$j=1(1)k-i+1$ in (7.d) and (7.e);

$$(8.a) \quad \gamma_{i,k}^* = \alpha_k^* \Gamma_{i,1} + (1 - \alpha_k^*) \tilde{\Gamma}_{i,1}; \quad i=0(1)k-1;$$

$$(8.b) \quad \gamma_{i+1,k} = \alpha_k \Gamma_{i,2} + (1 - \alpha_k) \tilde{\Gamma}_{i,2}; \quad i=0(1)k-1.$$

Then the (α_k, α_k^*) - VSVFM is based on the following formulae:

$$(9) \quad y_{n+k}^* = \alpha_k^* y_{n+k-1} + (1 - \alpha_k^*) y_{n+k-2} + \sum_{i=0}^{k-1} \gamma_{i,k}^* g_{n+k,i}^*;$$

$$(10) \quad y_{n+k} = \alpha_k y_{n+k-1} + (1 - \alpha_k) y_{n+k-2} + \sum_{i=1}^k \gamma_{i,k} g_{n+k,i}^*;$$

$$+ \gamma_{0,k}^* g_{n+k,0}^*.$$

ORDER $k=5$

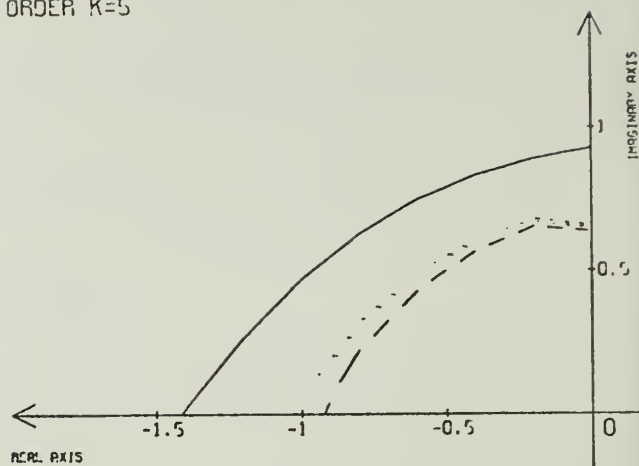


Fig. 1

(the differences $g_{n+k,i}^*$ in the sum in (10) can be found by the use of (6.b) with g replaced by g^*).

In our implementation many ideas from the most efficient codes ([1], [3], [4], [6]) were used. The code can be used with different values of α_k and α_k^* . By the choice $\alpha_k = \alpha_k^* = 1$, $k=2(1)12$, an Adams VSVFM may be obtained. This allows us to make comparisons in which only the use of different formulae in the VSVFM, but not the stepsize and order selection strategy, will have influence on the results.

Many problems from [3], [4] and [6] were run. As an example consider problem 12 in [4] with $\beta_1 = 4v$, $\beta_2 = 3v$, $\beta_3 = -v$, $\beta_4 = 0.001v$, $v = 1(1)12$, $\epsilon = 10^{-4}$, $[a, b] = [0, 50]$. The numbers of function evaluations (ND) are given in Table 2.

The results of the comparison between the Adams VSVFM's and the (α_k, α_k^*) VSVFM's can be summarized as follows:

(i) The (α_k, α_k^*) VSVFM's require more storage and computational work per step. Moreover the error constants of (2) and (4) are larger than those of the corresponding Adams methods. Therefore it is not efficient to use these methods if the accuracy requirements are dominant. But note that very often the number of function evaluations was not larger (a careful comparison

v	Adams VSVFM	(a_k, a_k^*) VSVFM
1	289	227
2	459	349
3	629	471
4	801	597
5	871	719
6	1139	839
7	1309	963
8	1477	1085
9	1577	1210
10	1815	1332
11	1984	1452
12	2155	1576

Table 2

showed that this is so because higher orders were selected by the (a_k, a_k^*) methods).

(ii) If the absolute stability requirements are dominant then the (a_k, a_k^*) methods will normally use a smaller number of steps in the integration.

(iii) The zero-stability properties of the (a_k, a_k^*) methods are the same as those of the Adams methods, see Zlatev [8].

(iv) A one step predictor-corrector scheme (the Euler formula as a predictor and the trapezoidal rule as a corrector) is used in the starting stage and in the situations where the order should be dropped to one (as an example consider the case where discontinuities in the right-hand side of (1) appear for some values of x in $[a, b]$), see more details in [6].

Our main conclusion is that (a_k, a_k^*) methods can successfully be used instead of the Adams methods in the case where only the absolute stability requirements restrict the selection of large stepsizes.

REFERENCES

1. Gear, C. W.: Algorithm 407, DIFSUB for solution of ordinary differential equations. Comm. ACM 14, 185-190(1970)
2. Gear, C. W., Watanabe, W. S.: Stability and convergence of variable order multistep methods. SIAM J. Numer. Anal. 11, 1044-1058(1974)
3. Hull, T. E., Enright, W. H., Fellen, B. M., Sedgwick, A. E.: Comparing methods for ordinary differential equations. SIAM J. Numer. Anal. 9, 603-637(1972)
4. Krogh, F. T.: On testing a subroutine for numerical integration of ordinary differential equations. J. Assoc. Comput. Mach. 20, 545-562(1973)
5. Shampine, L. F.: Stiffness and non-stiff differential equations solvers. In: Numerische Behandlung von Differentialgleichungen (L. Collatz, ed.). Int. Series Numer. Math., Vol. 27, pp. 287-301. Basel, Switzerland: Birkhäuser 1975
6. Shampine, L. F., Gordon, M. K.: Computer solution of ordinary differential equations: The initial value problem. San Francisco, Calif.: Freeman 1975
7. Thomsen, P. G., Zlatev, Z.: Two-parameter families of predictor-corrector methods for the solution of ordinary differential equations. Institute for Numerical Analysis, Technical University of Denmark, Lyngby, Denmark. Report No. NI-77-08, 1977
8. Zlatev, Z.: Stability properties of variable stepsize variable formula methods. Numer. Math. 31, 175-182(1978)

SOFTWARE DEVELOPMENT FOR STABILITY ANALYSIS OF NONLINEAR DIFFERENTIAL SYSTEMS

R. Leonard Brown
Dept. of Applied Mathematics and Computer Science
University of Virginia
Charlottesville, VA 22903

1. PROBLEM STATEMENT - Since the exact solution of a system of m first order ordinary differential equations

$$y' = f(y, t), \quad y(t_0) = y_0 \quad (1)$$

can be numerically approximated at only discrete points, the comparison of analytic and numerical solutions to determine accuracy and stability can only be made at discrete values t_i , $i = 0, \dots, n$. Since most numerical methods attempt to keep the stepsize $h_i = t_{i+1} - t_i$ constant for as many steps as practical subject to accuracy and stability constraints, the study of local accuracy and stability can be carried out for constant h . In particular, define the analytic k -step solution sequence $Y(t_n) = (y(t_{n-k+1}), y(t_{n-k+2}), \dots, y(t_n))$ and the numerical k -step solution sequence to be $Y_n = (y_{n-k+1}, y_{n-k+2}, \dots, y_n)$ where y_i is the numerical approximation of the solution $y(t_i)$ of (1) and each of $y(t_n)$, y_n is an m vector. This paper will describe a software package of interconnected programs and subroutines which can be used to compare the stability properties of Y_n generated by a particular multistep method with the properties of $Y(t_n)$ for a particular nonlinear differential function $f(y, t)$. Fig. 1 shows a flow diagram of this package.

The usual analysis of the stability of (1) involves two concepts: the equilibrium solution and the domain of attraction. Define the equilibrium solution $z(t)$ to be the analytic solution of (1) where $z_0 = z(t_0)$ is chosen such that

$$z'(t_0) = f(z_0, t_0) = 0. \quad (2)$$

For autonomous equations, the solution is $z(t) = z_0$. The domain of attraction can be found if there exists a function $v(y) > 0$ for $y \neq 0$, $v(0) = 0$, such that $dv(y)/dt \leq 0$ in a domain D about the equilibrium $z(t)$. This region is the domain of attraction and, as t increases, the difference between any two solutions of (1) generated by different initial values w_0 and x_0 within D decreases with respect to v , i.e.

$$|v(w(t+h)) - v(x(t+h))| < |v(w(t)) - v(x(t))|. \quad \text{If the function } v(y) = y^T Q y \text{ is used, where } Q \text{ is a positive definite matrix chosen by known techniques [1],}$$

then $v(y)$ is $\langle y, y \rangle$ for the scalar product $\langle u, v \rangle = u^T Q v$.

In fig. 1, box 2 represents a routine which solves for the $m(m+1)/2$ unique values of Q by dividing $f(y, t)$ into its linear and nonlinear parts so that

$$f(y, t) = B y + g(y, t)$$

Then a symmetric (or Hermitian) Q is found such that

$$B^T Q + Q B = I$$

where I is the m by m identity matrix. While this routine has not yet been written, the linearization can be automated by finding the approximate Jacobian B at $t = 0$ using column by column finite differencing, requiring m evaluations of $f(y, t)$.

Box 3 of fig. 1 represents a routine which, given an initial guess z inside D will find either z_0 , the equilibrium at $t = 0$, or else z_0^* , the equilibrium solution in a plane specified by holding $m-2$ values constant, if such a restriction does not produce algebraic contradictions. This routine is also being written and is discussed in section 4 below. In the sequel, z_0 will represent either z_0 or z_0^* as appropriate.

2. COMPUTATION OF STABILITY REGIONS - A concept similar to the analytic domain of attraction is sought for the analytic and numeric solution sequences. Dahlquist [2] has devised a function

$$v_{G,k,h}(Y_n) = \sum_{i=1}^k \sum_{j=1}^k g_{ij} \langle y_{n-k+i}, y_{n-k+j} \rangle$$

for some positive definite matrix $G = (g_{ij})$, where $\langle u, v \rangle$ is the scalar product above. Liniger and Odeh [3] have considered how to choose G to obtain the nonlinear equivalent of A -stability, and Dahlquist [4] mentions in a recent report that a program to compute G has been written at Stanford. This is represented by box 4 of fig. 1.

To facilitate computational and analytic manipulations with respect to multistep methods, it has been shown [2,5] that the numerical solution

using k-step multistep formulas

$$0 = \sum_{i=0}^k a_i y_{n-i} + h b_i f(y_{n-1}, t_{n-1})$$

have stability properties similar to numerical solutions using one leg k-step formulas

$$0 = \sum_{i=0}^k a_i y_{n-i} + h f(\sum_{i=0}^k b_i y_{r-i}, \sum_{i=0}^k b_i t_{n-i})$$

if $\sum_{i=0}^k b_i = 1$. Software is being developed which, given a point within the resulting domain finds the boundary of the region D' where $V_{G,k,h}(Y_1) - V_{G,k,h}(Y_0) \leq 0$, which is the discrete analog of the domain of attraction where $v'(y_0) \leq 0$. This can be used to compute D' for either the analytic or the numerical solution sequence, provided the analytic sequences $Y(t_1)$ and $Y(t_0)$ are available. The k-step numerical solution sequence is also computed using y_0 and k-1 exact (analytic) previous points to compute y_1 using the one leg k-step method under consideration. Box 1 of fig. 1 represents a compiler module which transforms a specification of the differential function $f(y, t)$ into two FORTRAN subroutines S and F. $S(T, Y_0)$ returns the correct solution vector at $t = T$ given that Y_0 is the initial value at $t = 0$, and using techniques described in section 3 below. $F(T, Y)$ returns the derivative of Y at time $t = T$ given that $y(t) = Y$. Box 5 of fig. 1 represents working software which uses brute force search and then

bisection on up to 80 lines radiating from z_0 along a 2-dimensional plane, storing the result in an array STAR.

The stability region D'' such that $V_{G,k,h}(Y_0) \leq V^*$ can also be computed. V^* is $\max V_{G,k,h}(Y)$ over all Y on the boundary of D'. This corresponds to all initial values that generate solutions that remain inside D' after k steps. Box 6 of fig. 1 represents the working software that computes and stores up to 80 points of the boundary of D''. These points are computed using bisection between z_0 and the boundary of D' stored in array STAR. It has been shown [5] that, if a domain of attraction D exists for $v(y) = y^t Q y$ that is convex, then the boundaries of D' and of D'' can be computed using rays from some point within D''.

3. AUTOMATED ANALYTIC SOLUTIONS - Since the analytic solution is required for k-1 steps before y_0 , means of obtaining $y(t_{-k+1}), \dots, y(t_{-1})$ are needed. Translation software, box 1 of fig. 1, is being developed to manipulate the specification of a system of m non-linear first order ordinary differential equations to produce a FORTRAN subroutine which, given y_0 at $t = 0$ and some value t, outputs $y(t)$ using power series solution techniques provided t is in the radius of convergence of the power series. For example,

$$\begin{aligned} x' &= -x - 2yx^2 \sin(t) \\ y' &= 5x - y + y/(t+4) \end{aligned}$$

has power series $x(t) = \sum_{i=0}^{\infty} a_i t^i$, $y(t) = \sum_{i=0}^{\infty} b_i t^i$ defined recursively by $a_0 = x_0$, $b_0 = y_0$,

$$a_i = (-a_{i-1} - 2b_{i-1} + \sin(t) \sum_{j=0}^{i-1} a_j b_{i-1-j})/i,$$

$$b_i = (5a_{i-1} - (t_0+3)/(t_0+4)b_{i-1})/i.$$

The regions D, D', and D'' for the analytic solution with $q_{11} = 37/44$, $q_{22} = 16/44$, $q_{12} = q_{21} = 3/44$, $G = I$, $h = .1$, and $k = 2$ appear in fig. 2.

4. EQUILIBRIUM - Although any point inside D'' can be used to generate the boundary of D' and D'', if the equilibrium (2) is used then good results for the analytic case are insured. Software corresponding to box 3 of fig. 1 is being written to handle computation of z_0 , possibly using an n dimensional chord method starting with a value that is known to be inside D''. This is an application of the software described in section 2 since a message that $V(Y_1) - V(Y_0) > 0$ can be generated to indicate that the point being used as the origin of the rays used to find the region boundary is itself not in the domain of attraction. A special case occurs when, for an n-dimensional system, n-2 initial values are held constant and the equilibrium initial value for the other 2 variables is found. This is both computationally simpler and more practical since only 2-dimensional

regions can be displayed on most graphics equipment. Since this routine is not available in the current software package, a z_0 is picked using knowledge of the function $f(y, t)$ being investigated.

5. GRAPHICAL OUTPUT - All of the currently working software has been gathered as subroutines to an interactive control MAIN program which uses a simple command language (the first n positive integers) to perform the n available functions. For those subroutines not yet available, such as the computation of the equilibrium, a default (0) is provided and execution of the command will allow the user to input a usable value. The 0 command produces a list of available commands; each command prompts the user in its use.

The graphics package, using TEKTRONIX PLOT10 software connected to a Control Data CYBER 172, prints the left, right, lower, and upper bounds of the figure to be drawn, then allows the user to override these limits. This allows all figures to be drawn to the same scale for comparison. A hardcopy device can be used to save the figures. The example in section 6 was drawn using this package.

6. EXAMPLE - Consider the analysis of the complex equation $y' = Ly$ where $L = (\ln y_0)/h$, which has two equilibria, a stable one at $z_0 = 0$ and an

unstable one $z_0 = 1$, for $v(y) = y^t y$, the Euclidean distance from 0, and $G = 1$. The analytic solution sequence is $y_i = y_0^{i+1}$, so the analytic domain of attraction and stability region are both bounded by the unit circle, and the unstable equilibrium is on the boundary. Fig. 3 shows D' about 1 for the numerical sequences generated by the Euler predictor (dotted), the Euler predictor with one backward Euler corrector (dash), and the 2-step Adams explicit formula (solid). Only implicit formulas that are stable at ∞ have domains of attraction that include $z_0 = 0$. In this case, there is no stability region D'' since the circle of radius V^* about $y = 0$ is not inside D' .

7. ACKNOWLEDGEMENT - This work has been supported in part by grant NSG 1335 from NASA through its Langley Research Center in Hampton, VA.

REFERENCES

1. Lehnigk, S.H., Stability Theorems for Linear Motions, Prentice-Hall, Englewood Cliffs, N.J (1966).
2. Dahlquist, G., On Stability and Error Analysis for Stiff Nonlinear Problems, Report NA-7508, Dept. of Information Processing, Royal Institute of Technology, Stockholm (1975).

Fig. 2 - Nonlinear time dependent example

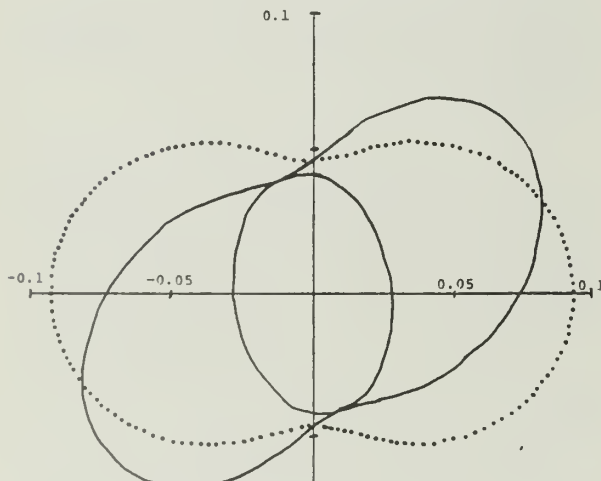
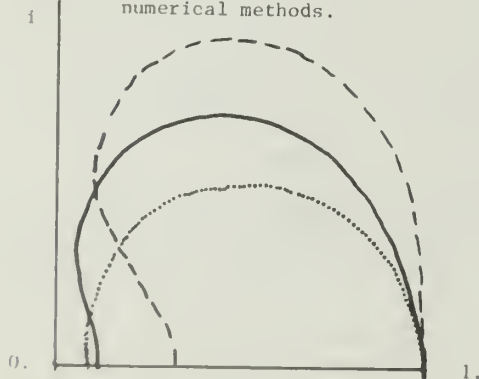
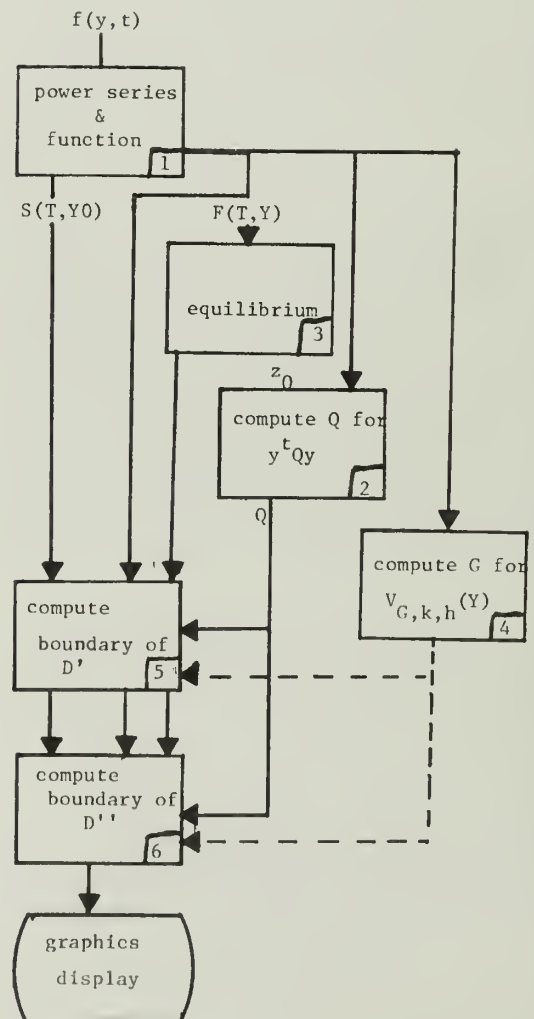


Fig. 3 - Linear example; D' for three numerical methods.



3. Liniger, W. And Odeh, F., On Liapunov Stability of Stiff Non-Linear Multi-step Difference Equations, AFOSR-TR-76-1023, IBM Thomas J. Watson Research Center (1976).
4. Dahlquist, G., G-stability is Equivalent to A-stability, Report NA-7805, Dept. of Information Processing, Royal Institute of Technology, Stockholm (1978).
5. Brown, R.L., Stability of Sequences Generated by Nonlinear Differential Systems, to appear, Math. Comp. (April, 1979).
6. Brown, R.L., Stability Analysis of Nonlinear Differential Sequences Generated Numerically, to appear, Int. Jnl. for Computers and Mathematics with Applications.

Fig. 1 - Flow graph of Software System.



SECOND DERIVATIVE EXTENDED BACKWARD DIFFERENTIATION FORMULAE FOR THE NUMERICAL INTEGRATION OF STIFF SYSTEMS

J. R. Cash
Department of Mathematics,
Imperial College,
South Kensington,
London S.W.7, England

In two recent papers the present author has introduced a class of extended backward differentiation formulae suitable for the approximate numerical integration of stiff systems of ordinary differential equations of the form

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0. \quad (1.1)$$

Extended backward differentiation formulae take the general form

$$\sum_{j=0}^k \bar{\alpha}_j y_{n+j} = h(\bar{\beta}_k f_{n+k} + \bar{\beta}_{k+1} f_{n+k+1}) \quad (1.2)$$

where the $\bar{\alpha}_j$'s and $\bar{\beta}_j$'s are constants, y_{n+j} is the approximate numerical solution obtained at x_{n+j} and $f_{n+j} =$

$f(x_{n+j}, y_{n+j})$. These formulae are similar in structure to the advanced multistep methods of Williams and de Hoog and to a subclass of the composite multistep methods considered by Bickart and his co-workers. However, the way in which the required solution of (1.2) is generated is entirely different from that considered by others. In an earlier paper it was shown that if these extended formulae are used in a certain precisely defined 'predictor-corrector' mode, with a conventional backward differentiation formula being used as a 'predictor' and (1.2) being used as the corrector, then it is possible to derive L-stable schemes of orders up to 4 and $A(\alpha)$ -stable schemes of orders up to 9. Numerical experiments with the resulting algorithm on a wide class of stiff test problems have indicated that it is often competitive with Hindmarsh's Gear Rev.3 which is widely accepted as being one of the most efficient general purpose codes for the integration of stiff systems currently available. Numerical experience has also shown that when integrating stiff systems of equations with jacobians having large complex eigenvalues lying very close to the imaginary axis using extended backward differentiation formulae it is

desirable that all of the basic integration formulae should be highly stable (not just A_0 -stable but $A(\alpha)$ -stable with α

close to $\pi/2$). This is because an algorithm employing $A(\alpha)$ -stable formulae with α small sometimes selects a high order formula with a small value of h to preserve stability rather than correctly choosing a more stable lower order formula with a larger value of h . The requirement that all of our formulae should be very highly stable leads us naturally to considering extended backward differentiation formulae employing second derivatives. These take the general form

$$y_{n+k} - y_{n+k-1} = h \sum_{j=0}^{k+1} \bar{\beta}_j f_{n+j} + h^2 \bar{\gamma}_k g_{n+k} \quad (1.3)$$

$$\text{where } g_{n+k} \equiv \left. \frac{df}{dx}(x, y(x)) \right|_{\substack{x=x_{n+k} \\ y=y_{n+k}}}$$

Note that we choose $\bar{\alpha}_j \equiv 0$, $j < k-1$ to ensure stability when $h = 0$.

Following the general strategy adopted for the implementation of (1.2) we use our formulae in a predictor-corrector mode with (1.3) being the corrector and a conventional second derivative formula of the form

$$y_{n+k} - y_{n+k-1} = h \sum_{j=0}^k \beta_j f_{n+j} + h^2 \gamma_k g_{n+k} \quad (1.4)$$

being used as the 'predictor'. This approach may be summarised in a step by step form in the following way:

- (1) Compute a quantity $\bar{y}_{n+k}^{(n)}$ as the solution of the conventional linear k -step formula

$$y_{n+k}^{-h\beta_k f_{n+k} - h^2 \gamma_k g_{n+k}} = y_{n+k-1} + h \sum_{j=0}^{k-1} \beta_j f_{n+j} \quad (1.5a)$$

(2) Compute $\bar{y}_{n+k+1}^{(n)}$ as the solution of

$$y_{n+k+1}^{-h\beta_k f_{n+k+1} - h^2 \gamma_k g_{n+k+1}} = \bar{y}_{n+k}^{(n)} + h\beta_{k-1} f(x_{n+k}, \bar{y}_{n+k}^{(n)}) + \sum_{j=0}^{k-2} h\beta_j f_{n+j+1} \quad (1.5b)$$

(3) Compute $\bar{f}_{n+k+1} \equiv f(x_{n+k+1}, \bar{y}_{n+k+1}^{(n)})$

(4) Compute y_{n+k} from (1.3) written in the form

$$y_{n+k}^{-h\bar{\beta}_k f_{n+k} - h^2 \bar{\gamma}_k g_{n+k}} = y_{n+k-1} + \sum_{j=0}^{k-1} h\bar{\beta}_j f_{n+j} + h\bar{\beta}_{k+1} \bar{f}_{n+k+1} \quad (1.5c)$$

We note that in implementing steps (1), (2) and (4) it is necessary in general to solve systems of nonlinear algebraic equations for the required solutions. In each case these are solved using a modified form of Newton iteration iterated to convergence. If the usual procedure of

neglecting the term $\frac{\partial^2 f}{\partial y^2}$ is adopted, the coefficient matrices associated with steps (1), (2) and (4) respectively are

$$I - h\beta_k J_{n+k} - h^2 \gamma_k J_{n+k}^2 \quad (A)$$

$$I - h\beta_k J_{n+k+1} - h^2 \gamma_k J_{n+k+1}^2 \quad (B)$$

and

$$I - h\bar{\beta}_k J_{n+k} - h^2 \bar{\gamma}_k J_{n+k}^2 \quad (C)$$

where J_{n+k} is some suitable approximation to the relevant jacobian matrix at x_{n+k} . Exhaustive comparisons conducted earlier for the case where $\gamma_k = 0$ have shown that if, with step (4), we use the coefficient matrix (A) rather than (C), then the iteration scheme in step 4 invariably converges if the iteration scheme in step (1) converges. Numerical experiments seem to indicate that this is also the case when $\gamma_k \neq 0$. Furthermore

it has been found that iteration scheme (2) normally converges if the coefficient matrix (A) is used. Thus in practice the only coefficient matrix which we need to compute and LU decompose is (A) and this leads to a substantial saving in computational effort.

It has been shown by Enright that for any integer k satisfying $1 \leq k \leq 7$ it is possible to derive stiffly stable schemes of the form (1.4) having orders $k+2$. It is straightforward to verify that for all positive integers k it is possible to derive formulae of the form (1.3) having order $k+3$. We now state the following lemma regarding the order of accuracy of the procedure defined by steps (1) - (4) above.

Lemma 1

Given that

- (1) Formula (1.5a) is of order $k+2$
- (2) Formula (1.3) is of order $k+3$
- (3) The implicit algebraic equations defining $\bar{y}_{n+k}^{(n)}$ and $\bar{y}_{n+k+1}^{(n)}$ are solved using an iteration scheme iterated to convergence, then scheme (1.5c) has order $k+3$.

This lemma is easy to verify by standard arguments.

Having defined our basic formulae we are now in a position to examine their regions of absolute stability. We do this by applying them to the usual scalar test equation $y' = \lambda y$, where λ is a complex constant with negative real part. It was found that these algorithms are very highly stable for $1 \leq k \leq 6$ and the corresponding regions of $A(\alpha)$ -stability are given in Table 1. Note in particular that these formulae are L-stable for orders up to and including 6 whereas Enright's conventional second derivative formulae are L-stable only up to order 4. If we consider still higher values of k we derive $A(\alpha)$ -stable formulae with the angle α decreasing to zero as k is increased. However, we have not investigated such formulae since, as stated previously, we are interested in deriving a package where all of the formulae are very highly stable. Furthermore, when solving stiff systems of equations it may not be desirable to have too many different order formulae as quite often a change in order results in the coefficient matrix of the Newton scheme having to be re-evaluated and LU decomposed and this is, of course, generally very expensive.

The exhaustive testing of a new algorithm is a long and complicated business and at the present time we are unable to draw any firm conclusions concerning the general efficiency of our algorithm. However, preliminary results obtained using a fixed order (and sometimes a fixed step) seem to indicate that the formulae discussed in the present paper are promising, especially when compared with conventional second derivative formulae. It is hoped that further research, particularly into choosing the

order and stepsize of the particular formula being used, will lead to the production of an efficient code incorporating the algorithms described in this note.

TABLE 1

% Order Region of $A(\alpha)$ -stability

1	4	90°
2	5	90°
3	6	90°
4	7	89°
5	8	87°
6	9	83°

COMPUTER STUDIES OF PLANETARY-TYPE EVOLUTION

Donald Greenspan
Department of Mathematics
University of Texas
Arlington, Texas 76019

John Collier
Computer Science Department
University of Wisconsin
Madison, Wisconsin 53706

ABSTRACT

In this paper a new, computer approach to the study of the interactions of particles with differing masses is applied to the study of planetary type evolution. The formulation contains an inherent self-reorganization property in which particles self-stratify in accordance with their masses. Computer examples are described and discussed.

1. INTRODUCTION

Recently [4], a new computer approach was developed for modelling the interactions of particles with differing masses. In this approach, matter is approximated by finite sets of particles, each of which is treated as an aggregate of molecules, and the active forces are appropriately rescaled. Inherent in the formulation is a natural, self-reorganization property, in which the particles self-stratify in accordance with their masses. After a precise mathematical and physical extension of the method, so as to include both long-range and short-range forces, we will apply it to the study of planetary-type evolution.

2. BASIC MATHEMATICAL DEFINITIONS AND FORMULAS

For positive time step Δt , let $t_k = k\Delta t$, $k = 0, 1, 2, \dots$. For $i = 1, 2, 3, \dots, N$, let particle P_i have mass m_i and at time t_k let P_i be located at $\vec{r}_{i,k} = (x_{i,k}, y_{i,k})$, have velocity $\vec{v}_{i,k} = (v_{i,k,x}, v_{i,k,y})$, and have acceleration $\vec{a}_{i,k} = (a_{i,k,x}, a_{i,k,y})$. Let position, velocity and acceleration be related by the "leap-frog" formulas ([3], p. 107):

$$\vec{v}_{i,1/2} = \vec{v}_{i,0} + \frac{\Delta t}{2} \vec{a}_{i,0} \quad (2.1)$$

$$\vec{v}_{i,k+1/2} = \vec{v}_{i,k-1/2} + (\Delta t) \vec{a}_{i,k}, \quad k = 1, 2, \dots \quad (2.2)$$

$$\vec{r}_{i,k+1} = \vec{r}_{i,k} + (\Delta t) \vec{v}_{i,k+1/2}, \quad k = 0, 1, 2, \dots \quad (2.3)$$

If $\vec{F}_{i,k}$ is the force acting on P_i at time t_k , where $\vec{F}_{i,k} = (F_{i,k,x}, F_{i,k,y})$, then we assume that force and acceleration are related by

$$\vec{F}_{i,k} = m_i \vec{a}_{i,k}. \quad (2.4)$$

Once an exact structure is given to $\vec{F}_{i,k}$, the motion of each particle will be determined recursively and explicitly by (2.1)-(2.4) from prescribed initial data. In the present paper, we will want $\vec{F}_{i,k}$ to incorporate both short-range and long-range components, and this will be implemented as follows. At time t_k , let $r_{ij,k}$ be the distance between P_i and P_j . Let G_{ij} (coefficient of molecular-type attraction), H_{ij} (coefficient of molecular-type repulsion), β_{ij} (exponent of molecular-type attraction), α_{ij} (exponent of molecular-type repulsion) and G_{ij}^* (gravitational constant) be constants determined by P_i and P_j , subject to the constraints (see [6]): $G_{ij} \geq 0$, $H_{ij} \geq 0$, $\alpha_{ij} \geq \beta_{ij} \geq 2$, $G_{ij}^* \geq 0$. Then the force $(\vec{F}_{i,k,x}, \vec{F}_{i,k,y})$ exerted on P_i by P_j is defined by

$$\vec{F}_{i,k,x} = \left[\frac{-G_{ij}}{r_{ij,k}^{\beta_{ij}}} + \frac{H_{ij}}{r_{ij,k}^{\alpha_{ij}}} - \frac{G_{ij}^*}{r_{ij,k}^2} \right] \frac{(m_i m_j)(x_{i,k} - x_{j,k})}{r_{ij,k}} \quad (2.5)$$

$$\vec{F}_{i,k,y} = \left[\frac{-G_{ij}}{r_{ij,k}^{\beta_{ij}}} + \frac{H_{ij}}{r_{ij,k}^{\alpha_{ij}}} - \frac{G_{ij}^*}{r_{ij,k}^2} \right] \frac{(m_i m_j)(y_{i,k} - y_{j,k})}{r_{ij,k}} \quad (2.6)$$

The total force $(F_{i,k,x}, F_{i,k,y})$ on P_i due to all the other $(N - 1)$ particles is given by

$$F_{i,k,x} = \sum_{\substack{j=1 \\ j \neq i}}^N \vec{F}_{i,k,x}; \quad F_{i,k,y} = \sum_{\substack{j=1 \\ j \neq i}}^N \vec{F}_{i,k,y} \quad (2.7)$$

The formulation (2.1)-(2.7) is explicit and economical though nonconservative. Conservation of energy and momenta can be achieved [3], but

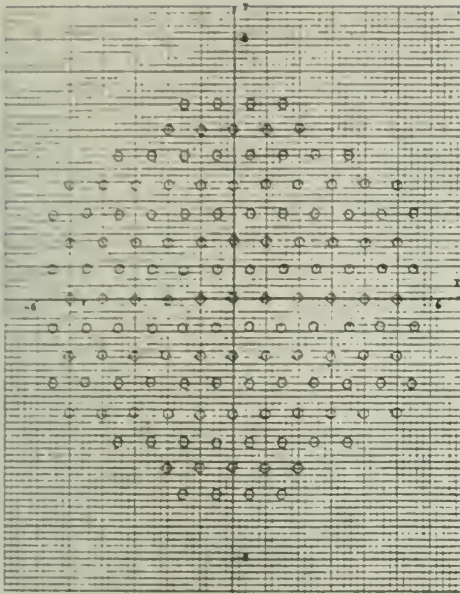


Fig. 1.

only through an implicit, less economical approach. Throughout, the time step to be used in (2.1)-(2.3) will always be $\Delta t = 10^{-4}$ and a comprehensive FORTRAN program for implementation of (2.1)-(2.7) is given in the Appendix of [5]. It should be noted that the way in which (2.1)-(2.7) will be applied lends itself directly to parallel computation also [2].

3. BASIC PHYSICAL ASSUMPTIONS AND DEFINITIONS

We will consider a system of 137 particles, so that $N = 137$. This parameter was determined solely by economic constraints. The initial particle positions will always be those shown in Figure 1.

Next, while in motion we will require a rule for determining the physical state of each particle. Such matters can be exceedingly complex [1] so that, for the present, we will use, as in [5], the following intuitive notion. We will use "temperature" as a phenomenon of a particle's "local" velocity, that is, its velocity relative to neighboring particles. Thus, when a particle is rotating within a large system, the gross system velocities should have no effect on the particle's temperature.

As a last consideration, we will allow for radiation into and out of the system. This will be accomplished as follows. Consider the vertical strip regions

$$k\gamma \leq x \leq (k+1)\gamma; \quad \gamma > 0,$$

$$k = 0, \pm 1, \pm 2, \pm 3, \dots$$

In each strip, the particles with maximum and minimum y coordinates are called outer particles. An outer particle will be called a light-side particle, that is, it faces a sun, if $y_{i,k} \geq y_k$.

Otherwise, it is called a dark-side particle. Light-side particles will receive radiant heat while dark-side particles will emanate radiant heat. This is implemented as follows. Every k^* steps, outer particle velocities are reset by

the rule

$$\text{light-side particle: } \vec{v}_{i,k} \rightarrow 1.001 \vec{v}_{i,k} \quad (3.1)$$

$$\text{dark-side particle: } \vec{v}_{i,k} \rightarrow 0.9955 \vec{v}_{i,k}.$$

4. EXAMPLES

A variety of examples were run with various combinations of parameter choices selected from $\theta = 4, 5$; $\epsilon = 5, 10$; $k^* = 50, 100$; $D = 2.1, 2.3$; $\gamma = 0.1, 1.0, 2.0$; and $G_{ij}^* = 0.01, 0.001$. From these we will first describe two, in both of which the parameter choices are $\theta = 4$, $\epsilon = 10$, $D = 2.3$, $k^* = 100$, $\gamma = 1$, and $G_{ij}^* = 0.001$.

The results of the variety of other examples run can only be summarized easily as follows. Only minor changes in input parameters often resulted in dramatic changes in dynamical behavior [4]. This is, of course, consistent with the diversity one observes among the planets and moons of the solar system.

Other computations, aimed at duplicating the development of galaxy arms, all ended in failure. The systems invariably self-reorganized into one or more relatively elliptic or circular subsystems. This suggests to us the strong possibility that more than gravitation is involved in galaxy arm formation.

Finally, note that initial calculations with a 239 particle configuration were begun, but these had to be discontinued due to a lack of funds.

REFERENCES

1. BARKER, J.A. and HENDERSON, D.: "What is 'liquid'? Understanding the state of matter", Rev. Mod. Phys., 48, 1976, pp. 587-671.
2. FITZWATER, D.R.: "The Formal Design and Analysis of Distributed Data-Processing Systems", TR322, Dept. Comp. Sci., U. Wis., Madison, 1978.
3. GREENSPAN, D.: Discrete Models, Addison-Wesley, Reading, Mass., 1973.
4. GREENSPAN, D.: "Computer studies of interactions of particles with differing masses", Jour. Comp. and Appl. Math., 3, 1977, pp. 145-154.
5. GREENSPAN, D. and COLLIER, J.: "Computer studies of swirling particle fluids and the evolution of planetary-type bodies", TR 299, Dept. Comp. Sci., U. Wis., Madison, 1977.
6. HIRSCHFELDER, C.F.; CURTISS, C.F. and BIRD, R.B.: Molecular theory of gases and liquids, Wiley, N.Y., 1954.

On the Reliability of Error Estimators for Explicit Runge-Kutta Procedures

J.H. Verner
Department of Mathematics and Statistics
Queen's University, Kingston, Canada

Abstract. Arbitrary parameters for $(s, p-1(p))$ Runge-Kutta procedures should be selected so that resulting codes yield reliable results efficiently. Shampine suggests as a compromise a process for selecting parameters which attempts to optimize efficiency subject to a constraint designed to provide reliability. This process is adapted to choose parameters for procedures with $p = 5, 6$, and some comparisons with known procedures are given.

1. Introduction

Currently $(s, p-1(p))$ -procedures are the most efficient Runge-Kutta processes available. Pairs of formulas of successive orders of accuracy provide, for example, an approximation of order $p-1$ and an error estimate (EE) of order p to the local error (LE). It is known that families of these procedures may be represented in terms of arbitrary parameters, yet there is little agreement on how they should be selected.

Production codes are implemented so that some function of EE is bounded by a user specified tolerance. A basic assumption in using such a code is that the behaviour of the global error is reflected by that of the local error. In showing that this occurs for various strategies, Shampine [3] needs to assume that the actual local error is bounded by the specified tolerance. Accordingly, parameters should be selected so that EE reflects the behaviour of LE. A second more prevalent criterion for selection is that of minimizing coefficients of the principal local truncation error in order to increase efficiency. To enhance both accuracy and efficiency Shampine [2] proposes as a compromise, a constrained

optimization of the parameters. This process is modified below in order to make the computation feasible for higher order formulas.

Thus we propose to derive reliable ($|LE-EE|$ is small), efficient ($|EE|$ is small), accurate ($|LE|$ is small) procedures. Because the optimization cannot be viewed as minimizing these quantities, the resulting formulas cannot be considered optimal. While potentially useful formulas may be derived, the analysis is more appropriate for indicating which procedures should be avoided.

2. Selection Criteria

For elementary differentials $\{D_{qi}\}$ evaluated at the current approximation, error expressions in the next step are

$$LE = h^p \sum_{i=1}^N \frac{1}{p!} T^{p-1}_{Di} + \sum_{q=p+1}^{\infty} h^q \sum_{i=1}^N \frac{1}{q!} T^{p-1}_{Di} \frac{1}{q!}$$

and

$$EE = h^p \sum_{i=1}^N \frac{1}{p!} T^{p-1}_{Di} + \sum_{q=p+1}^{\infty} h^q \sum_{i=1}^N \frac{1}{q!} (T^{p-1}_{Di} - T^p_{Di}) \frac{1}{q!}$$

where \underline{T}^{p-1} and \underline{T}^p are vectors of numerical constants determined by the methods of orders $p-1$ and p respectively. While for h small, there will be little "contamination" of the error estimate, efficient solutions for moderate tolerances will require moderate to large stepsizes. Hence, the contamination may be substantial, for example, if any coefficients $\{T_{pi}^{p-1}\}$ are zero, or if the magnitude of \underline{T}^p is substantially larger than that of \underline{T}^{p-1} .

These comments indicate that the reliability and efficiency of a procedure may be assessed by considering the parameters

$$\alpha = \max_i |T_{pi}^{p-1}|,$$

$$\beta_q = \max_i |T_{qi}^{p-1}| = \max_i |T_{qi}^{p-1}|, \quad q > p,$$

$$\gamma_q = \max_i |T_{qi}^{p-1}| = \max_i |T_{qi}^p|, \quad q > p,$$

$$\delta_q = \max_i \left| \frac{T_{qi}^{p-1} - T_{qi}^p}{T_{qi}^{p-1}} \right|, \quad q > p,$$

$$z_q = \text{number of coefficients } T_{qi}^{p-1} \equiv 0.$$

Following the intent of Shampine's proposal, new procedures are derived to minimize α subject to making $\beta_{p+1}, \gamma_{p+1}$ small (≤ 1 if possible) and $z_p = 0$. (Even this optimization may not yield reliable procedures, for example, if $T_{qi}^{p-1} = T_{qi}^p$ for some indices.)

3. Derivation and Comparison

A four-parameter family of $(6,4(5))$ procedures is known [5]. If the nodes are confined to $[0,1]$ and $\beta_6 \leq 1$, α is minimized if $c_3 = \frac{4+\sqrt{6}}{5}$. A choice of c_5 so that $\beta_6 = 1$

minimizes δ_6 , and a further choice of c_6 minimizes γ_6 . The procedure V3 has nodes $(0, \frac{1}{4}, \frac{4+\sqrt{6}}{5}, \frac{4-\sqrt{6}}{10}, \frac{18-\sqrt{6}}{18}, \frac{96-11\sqrt{6}}{90})$. This is contrasted to other procedures: F - Fehlberg [1], V1 [4], V2 [5]. The fact

Procedure	z_5	α	β_6	γ_6	δ_6
F1	0	2.08(-3)	1.67	.20	1.92
F2	0	1.28(-3)	2.59	.55	.46
V2	0	8.57(-4)	3.77	.67	.36
V2	5	6.25(-4)	4.44	1.00	.62
V3	0	1.87(-3)	1.00	.33	1.16

Table 1: Parameters for $(6,4(5))$ procedures

Procedure	FCN	STEPS	MAX ERR	FRAC DEC
F1	134925	21970	1.3	.002
F2	120406	19541	10.4	.022
V2	110803	17903	10.6	.039
V2	Disaster for $y' = -y$			
V3	147757	24157	1.8	.002

Table 2: DETEST for $(6,4(5))$ -procedures

that V3 does not perform as well as F1 on DETEST indicates the deficiencies of this analysis.

To obtain an $(8,5(6))$ procedure with $\beta_7 \leq 1$ is possible only for $\alpha \approx 9.2(-4)$. As such a procedure was unlikely to be as efficient as DVERK, others were considered. Without constraining β_7 , α is minimized by the

nodes $(0, \frac{4}{25}, \frac{8}{25}, \frac{20}{21}, \frac{9}{10}, 1, \frac{1}{15}, 1)$ to the

value $1.98(-4)$; however $\beta_7 = 6.15$. As a more reliable alternate to DVERK we suggest the procedure (V3) with nodes

$(0, \frac{1}{6}, \frac{10-2\sqrt{10}}{15}, \frac{5+\sqrt{10}}{15}, \frac{4}{3}, 1, \frac{1}{15}, 1)$.

Observe that the results of Table 4 are supportive of the indicators of Table 3. For $p = 7, 8$, similar comparisons are not so consistent: while results from DETEST indicate procedures of [5] are reliable and efficient, the parameters β, γ, δ , do not adequately

distinguish them from other known procedures.

Procedure	z_6	α	β_7	γ_7	δ_7
F	14	4.62(-4)	2.07	1.00	3.50
DVERK	5	4.62(-4)	4.61	.91	3.50
V1	0	1.69(-3)	.90	.15	1.40
V2	5	4.62(-4)	1.43	.69	8.01
V3	0	3.67(-4)	1.54	.60	2.85

Table 3: Parameters for (8,5(6)) procedures

Procedure	FCN	STEPS	MAX ERR	FRAC DEC
F	75091	8913	22.3	.063
DVERK	73206	8603	15.3	.134
V1	92139	10995	2.5	.002
V2	80282	9568	10.7	.022
V3	76184	9012	4.1	.022

TABLE 4: DETEST for (8,5(6))-procedures

In conclusion, we can only advise that procedures with $z_p \neq 0$, or any of β , γ , δ relatively large should be avoided.

Acknowledgements: This research was supported by the National Research Council of Canada under grant A8147.

REFERENCES

1. W.H. Enright, R. Bedet, I. Farkas, T.E. Hull, Tech. Report 68, 1974, Univ. of Toronto
2. L.F. Shampine, Report SAND 76-0370, 1976, Sandia Laboratories
3. L.F. Shampine, Appl. Math. and Comp. 3, 189-210, 1977.
4. J.H. Verner, SIAM J. on N.A. 15, 772-790, 1978.
5. J.H. Verner, Preprint 1978-19, 1978, Queen's Univ.

A COMPARISON BETWEEN SEVERAL PIECEWISE CONTINUOUS
ONE STEP INTEGRATION TECHNIQUES

J. P. HENNART AND R. ENGLAND

Universidad Nacional Autónoma de México
IIMAS (Institute for Research in Applied Mathematics and Systems)
Apartado Postal 20-726, México 20, D. F. (México)

In a series of papers by the first author [1-3], a class of one-step schemes has been developed which is particularly well adapted to the numerical integration of stiff systems of ODE'S. These schemes lead to piecewise continuous approximations in time with the added possibility of exponential fitting, and provide therefore a very general framework for the development of finite elements in time in connection with parabolic evolution problems [4,5]. In this paper, details of some of these schemes are given, for cases which do not require second derivative evaluation, and they are compared with similar alternative schemes providing piecewise polynomial solutions on the one hand [6,7], or exponential fitting on the other [8-10].

Given a system of n ODE'S in autonomous form:

$$\dot{\underset{\sim}{y}} = \underset{\sim}{f}(\underset{\sim}{y}(t)), \quad (1)$$

one gets after integration over $[t_i, t_{i+1}]$:

$$\underset{\sim}{y}(t_{i+1}) = \underset{\sim}{y}(t_i) + \int_{t_i}^{t_{i+1}} \underset{\sim}{f}(\underset{\sim}{y}(t)) dt. \quad (2)$$

In the integral on the righthand side, $\underset{\sim}{y}(t)$ is then approximated by its two-point Taylor interpolant $\underset{\sim}{y}_{pq}(t)$, leading to

$$\underset{\sim}{y}_{i+1} = \underset{\sim}{y}_i + \int_{t_i}^{t_{i+1}} \underset{\sim}{f}(\underset{\sim}{y}_{pq}(t)) dt, \quad (3)$$

where p, q are nonnegative integers, while each component of $\underset{\sim}{y}_{pq}$ belongs to the linear space

$$S \equiv \{t^n \exp(\lambda_m t) \mid n=0, \dots, N_{m-1}; m=1, \dots, M; \sum_m N_m = p+q\}, \quad (4)$$

where the λ_m 's are given distinct real numbers, one of them usually being zero. $\underset{\sim}{y}_{pq}$ is then chosen to satisfy the $p+q$ interpolation conditions

$$\underset{\sim}{y}_{pq}^{(r)}(t_i) = \underset{\sim}{y}_i^{(r)}, \quad r=0, \dots, p-1, \quad (5.a)$$

and

$$\underset{\sim}{y}_{pq}^{(s)}(t_{i+1}) = \underset{\sim}{y}_{i+1}^{(s)}, \quad s=0, \dots, q-1, \quad (5.b)$$

the unknown parameters $\underset{\sim}{y}_i^{(r)}$, $r=1, \dots, p-1$, and $\underset{\sim}{y}_{i+1}^{(s)}$, $s=1, \dots, q-1$, being finally eliminated by multiple collocation at $t=t_i$ and $t=t_{i+1}$:

$$\underset{\sim}{y}_{pq}^{(r)}(t_i) = \underset{\sim}{f}^{(r-1)}(\underset{\sim}{y}_{pq}(t_i)), \quad r=1, \dots, p-1, \quad (6.a)$$

and

$$\underset{\sim}{y}_{pq}^{(s)}(t_{i+1}) = \underset{\sim}{f}^{(s-1)}(\underset{\sim}{y}_{pq}(t_{i+1})), \quad s=1, \dots, q-1, \quad (6.b)$$

with $\underset{\sim}{y}_{pq}(t_i)$ known either from the initial conditions ($i=0$) or from integration over the previous interval $[t_{i-1}, t_i]$. After such a substitution, Eq. (3) becomes:

$$\underset{\sim}{y}_{i+1} = \underset{\sim}{y}_i + \int_{t_i}^{t_{i+1}} \underset{\sim}{f}(\underset{\sim}{y}_{pq}(\underset{\sim}{y}_i, \underset{\sim}{y}_{i+1}, t)) dt, \quad (7)$$

i.e. a system of nonlinear equations in the unknowns $\underset{\sim}{y}_{i+1}$. Eq. (7) defines a class of discrete one-step integration methods, depending on the quadrature formula which is used to approximate the integral, in addition to the choice of p and q . Assuming that $\underset{\sim}{y}$ exists and is unique, each of its components depends on $(p+q)$ parameters which are the corresponding components of $\underset{\sim}{y}^{(r)}$, $r=0, \dots, p-1$ (if $p \geq 1$), and of $\underset{\sim}{y}_{i+1}^{(s)}$, $s=0, \dots, q-1$ (if $q \geq 1$). Explicitly

$$\underset{\sim}{y}_{pq}(t) = \sum_{r=0}^{p-1} \underset{\sim}{y}_i^{(r)} u_r \left(\frac{t-t_i}{h} \right) \frac{h^r}{r!} + \sum_{s=0}^{q-1} \underset{\sim}{y}_{i+1}^{(s)} v_{i+1} \left(\frac{t-t_i}{h} \right) \frac{h^s}{s!}, \quad (8)$$

where $h = t_{i+1} - t_i$ and the functions $u_r, v_s \in S$ satisfy

$$u_r^{(\ell)}(0) = \delta_{r\ell}, \quad r, \ell=0, \dots, p-1, \quad (9.a)$$

$$u_r^{(\ell)}(1) = 0, \quad r, \ell=0, \dots, q-1, \quad (9.b)$$

and

$$v_s^{(\ell)}(0) = 0, \quad s, \ell=0, \dots, p-1, \quad (9.c)$$

$$v_s^{(\ell)}(1) = \delta_{s\ell}, \quad s, \ell=0, \dots, q-1, \quad (9.d)$$

so that (5) is automatically satisfied. Then, when

some k point quadrature formula is used, Eq. (7) becomes

$$y_{i+1} = y_i + h \sum_{\ell=1}^k w_{\ell} f(y_{\ell}^{pq}(t_i + \theta_{\ell} h)) \quad (10)$$

with $0 \leq \theta_1 < \dots < \theta_k \leq 1$, that is

$$y_{i+1} = y_i + h \sum_{\ell=1}^k w_{\ell} f \left\{ \sum_{r=0}^{p-1} y_i^{(r)} u_r(\theta_{\ell}) \frac{h^r}{r!} + \sum_{s=0}^{q-1} y_{i+1}^{(s)} v_{i+1}(\theta_{\ell}) \frac{h^s}{s!} \right\}. \quad (11)$$

Restricting attention to the case $p, q \leq 2$ with $S = \Pi^{p+q-1}$, the space of polynomials of degree less than or equal to $(p+q-1)$, (11) leads to a class of $(k+2)$ stage Runge-Kutta schemes which can be characterized by the following array in Butcher's notation (and with $p=q=2$):

0	0	0	...	0	0
θ_1	$u_1(\theta_1)$	$w_1 v_0(\theta_1)$...	$w_k v_0(\theta_1)$	$v_1(\theta_1)$
.
.
.
θ_k	$u_1(\theta_k)$	$w_1 v_0(\theta_k)$...	$w_k v_0(\theta_k)$	$v_1(\theta_k)$
1	0	w_1	...	w_k	0
	0	w_1	...	w_k	0

The special cases $\theta_1 = 0$ and (or) $\theta_k = 1$, and those with p and (or) $q < 2$, are easily derived from this general class by noting the properties (9) of the functions u_r, v_s .

A number of particular cases are of interest, among which the following may be mentioned.

$p=1, q=0$, and any quadrature rule exact for a constant integrand, gives the forward Euler method.

$p=0, q=1$, and any quadrature rule exact for a constant integrand, gives the backward Euler method.

$p=2, q=0$, and any two-point quadrature rule exact for a linear integrand, gives a 3 stage, second order, explicit Runge-Kutta method. The family with $\theta_1=0$ gives all the 2 stage methods of that type.

$p=q=1$, and any two-point quadrature rule exact for a linear integrand, gives a 2 stage, second order, implicit Runge-Kutta method. The family with $\theta_1=0$ consists of semi-explicit methods, with Butcher's array:

0	0	0
2	$\frac{1}{2} - \frac{1}{2}$	$\frac{1}{2}$
	$1 - \frac{1}{2}$	$\frac{1}{2}$

which include the well known cases: $\theta_2 = \frac{1}{2}$ (implicit mid point method), $\theta_2 = 1$ (trapezium method).

$p=0, q=2$, and any two-point quadrature rule exact for a linear integrand, gives a 3 stage, second order, implicit Runge-Kutta method. The family with $\theta_2 = 1$ consists of 2 stage methods, but these show no apparent advantage, in terms of effort or stability, over methods of third order with $p=1, q=2$.

$p=2, q=1$, and any three-point quadrature rule exact for a quadratic integrand, gives a 4 stage, third order, implicit Runge-Kutta method. The case with $\theta_1 = 0, \theta_2 = 2/3$ is a 2 stage, semi-explicit, Radau method, of higher order than those given by (13), but of limited interest because it has a bounded stability region.

$p=1, q=2$, and any three-point quadrature rule exact for a quadratic integrand, gives a 4 stage, third order, implicit Runge-Kutta method. The family with $\theta_1 = 0, \theta_2 = 1$ consists of 3 stage methods, amongst which should be mentioned the Radau and Lobatto methods obtained with

$\theta_2 = 1/3, 1/2, 2/3$ having Butcher's arrays: (20)

1/3	5/12	-1/12	0	0	0	0	0	0	0
1	3/4	1/4	1/2	1/8	1/2-1/8	2/3	2/9	2/3	-2/9
	3/4	1/4	1	1/6	2/3	1/6	1	1/4	3/4
				1/6	2/3	1/6		1/4	3/4

the first showing possible advantages both for storage and convergence during solution of the nonlinear equations (7) at each step.

$p=q=2$, and any three-point quadrature rule exact for a cubic integrand, gives a 5 stage, fourth order, implicit Runge-Kutta method. The case with $\theta_1=0, \theta_2=1$ is a 3 stage, Lobatto method with Butcher's array:

0	0	0	0
1/2	5/24	1/3	-1/24
1	1/6	2/3	1/6
	1/6	2/3	1/6

an A-stable method which has been given by other authors.

At this point, it is interesting to point out that the "optimal" Runge-Kutta schemes based on Gauss-Legendre quadrature [6] do not appear as special cases of the above formalism. These optimal schemes are more finite difference oriented in the sense that they achieve superconvergence at the mesh points. However, as is well known, they lose accuracy between them. Our schemes, which were studied in detail in [1] for general p and q , do not present that disadvantage: order $p+q$ is achieved not only at the mesh points but also between them, under the only assumption that the quadrature rule is exact when applied to the members of S . The basic idea for developing such a formalism was to provide a piecewise continuous representation in time that could match in a fully consistent way the high order accuracies offered by finite elements in space for parabolic evolution problems.

It turns out that in the polynomial case, the stability properties only depend on p and q (if the quadrature scheme is of sufficient accuracy): namely the schemes with $p=q$ are A-stable, while the schemes with $q=p+1$, $p+2$ are strongly A-stable.

In the general case where exponential basis functions are present (Eq. (4)), it is possible to show [5] that the above schemes when applied to the test equation $\dot{y} = \lambda y$ are equivalent to the multi-point Padé approximant schemes interpolating the exponential N times at $x=h\lambda_m$, $m=1, \dots, M$, and at least once at $\lambda=0$. These schemes have been studied and compared with other exponentially fitted schemes [8-10] in reference 11 in relation with systems of nonlinear ODE's and some of them applied in reference 12 to the nuclear reactor point kinetics equations. To be applicable, they required that the equations (if nonlinear) be linearized, and that mean values of the parameters be taken: both approximations introduce third order local errors, and the corresponding schemes were therefore limited to second order. This is not the case for the new schemes presented here which retain order $p+q$ (or at least $p+q-j$, if j is the number of interpolation points that remain far from the origin) for any problem. As far as stability is concerned, all the results previously obtained for the multi-point Padé approximants are valid here [8-12].

For a general space S of approximants as given in (4), it is again required that the quadrature rule be exact for its members. In the presence of exponential basis functions, this problem is less documented than in the polynomial case, although it is known that Gauss-type formulae exist for such Chebyshev systems [13]. Since very few realistic problems would require fitting of more than one exponential node, an attractive alternative may be described as follows. First multiply both members of (1) by $z(t)$ to get

$$z \dot{y} = z f(y(t)), \quad (22)$$

or equivalently

$$\frac{d}{dt}(zy) = \dot{z}y + z f(y(t)). \quad (23)$$

If (23) instead of (1) is integrated over $[t_i, t_{i+1}]$ and y replaced as above by y^{pq} in the right hand side of the resulting expression, it is easy to realize that if $z(t)$ is chosen to be proportional to $\exp(-\lambda t)$, $\lambda \neq 0$, the components of y behaving as $\exp(\lambda t)$ will be integrated exactly independently of S (and of the associated quadrature rule).

Apart from

- (i) a family of schemes for advancing the solution of Eq. (1) from t_i to t_{i+1} , a practical implementation also requires:
 - (ii) a method for estimating the local truncation error;
 - (iii) a strategy for selecting the step length h , and possibly also the order or particular member of the family to be used.
- In addition, any implicit scheme involves the solu-

tion of a system of nonlinear equations such as (7), and the implementation also requires:

- (iv) an algorithm for solving a system of nonlinear equations, while if output is required at points much more closely spaced than the grid points used for the integration;
- (v) an interpolation process should ideally be provided. Some comparisons have been performed on complete implementations, without separating the five aspects mentioned above [14]. However, in relation with this work, a modular structure is being developed with each aspect treated in a separate routine, thus making it easy to keep the elements (ii), (iii), (iv) constant, and vary the schemes (i) used - (v) depending upon (i). It will also facilitate a later study of the effects of varying (ii), (iii) or (iv).

Detailed run statistics will be reported later.

REFERENCES.

1. J. P. Hennart, "One-step piecewise polynomial multiple collocation methods for initial value problems", Math. Comp. 31, 24-36 (1977).
2. J. P. Hennart, "Piecewise polynomial approximations for nuclear reactor point and space kinetics", Nucl. Sci. and Engng. 64, 875-882 (1977).
3. J. P. Hennart, "Some recent one-step formulae for integrating stiff systems of ODE's", Séminaire d'Analyse Numérique No. 253, Université Scientifique et Médicale de Grenoble (1976).
4. J. P. Hennart, "Multiple collocation finite elements in time for parabolic evolution problems", to appear in The Mathematics of Finite Elements and Applications III-MAFELAP 1978, J. R. Whiteman, Ed., Academic Press, London.
5. J. P. Hennart, "Finite elements in time and space for transient reactor neutronics analysis", to appear in Proceedings of the National Topical Meeting on Computational Methods in Nuclear Engineering, American Nuclear Society (1979).
6. B. L. Hulme, "Discrete Galerkin and related one-step methods for ordinary differential equations", Math. Comp. 26, 881-891 (1972).
7. K. Wright, "Some relationships between implicit Runge-Kutta, collocation and Lanczos τ methods, and their stability properties", BIT 10, 217-227 (1970).
8. W. Liniger and R. A. Willoughby, "Efficient integration methods for stiff systems of ordinary differential equations", SIAM J. Numer. Anal. 7, 47-66 (1970).
9. B. L. Ehle and Z. Picel, "Two-parameter, arbitrary order, exponential approximations for stiff equations", Math. Comp. 29, 501-511 (1975).
10. F. H. Chipman, "A note on implicit A-stable R-K methods with parameters", BIT 16, 223-227 (1976).

11. J. P. Hennart, "The multipoint Padé approximant approach to exponentially fitted one-step formulae", Comunicaciones Técnicas, Vol. 6, Serie B, No. 119, CINAS-UNAM, México (1975).
12. J. P. Hennart and B. Torres Barrios, "Generalized Padé and Chebyshev approximations for point kinetics", Nucl. Sci. and Engng. 59, 170-187 (1976).
13. S. Karlin and W. J. Studden, Tchebycheff Systems: With Applications in Analysis and Statistics, John Wiley & Sons, New York (1966).
14. W. H. Enright, T. E. Hull, and B. Lindberg, "Comparing numerical methods for stiff systems of O.D.E.'s, BIT 15, 10-48 (1975).

ON THE A-STABILITY OF NUMERICAL MULTISTEP

METHODS FOR SOLVING IVPs IN ODEs

BY

HASSAN NASR

Department of Mathematics, Military Technical College

Cairo 1979

Abstract:

It is well known (Dahlquist 1963) that for the study of the A-stability of linear Multistep Methods for solving IVPs in ODEs, A-stable method for order higher than two do not exist. This paper gives the necessary and sufficient condition for the general multistep methods to be A-stable. It includes an example for a new A-stable method of order 4.

Introduction :

Let us recall the definition of the General Blocking Over-Implicit Multistep (GBOM) Method, by Hassan Nasr in Aplikace Matematiky (under printing). The method is used for solving the ODE

$$y' = f(x, y), \quad x \in [a, b] \quad (1)$$

with the initial condition

$$y(a) = c \quad (2)$$

where the function $f(x, y)$ is defined, continuous and satisfying the Lipschitz conditions w.r.t. y in $[a, b] \times (-\infty, \infty)$ so that the existence and uniqueness of the solution of equation (1) under the initial condition (2) be guaranteed in the whole $[a, b]$. The essence of this method lies in the fact that in one step of the method, the approximate solution is computed simultaneously in k points supposing that it is known in s points ($1 \leq s \leq k$). Let the points $x_{jk} = a + jmh$, $j = 0, 1, \dots$ where m is a

positive integer and $h > 0$ is the integration step, will be called basic points and the points $x_{jk+i} = x_{jk} + w_i h$,

$i = 1, \dots, k-1$ where w_i are any real

numbers, will be called intermediate points while the approximate solution in the point x_{jk+i} ($i = 0, \dots, k-1$,

$j = 0, 1, \dots$) will be denoted by $\tilde{y}(x_{jk+i})$,

the s -dimensional vector

$\underline{z}_j = (x_{jk}, \dots, x_{(j+1)k+s-1})$ and the s -dimensional

vector of the approximate solution at these points by $\tilde{y}(\underline{z}_j) = (\tilde{y}(x_{jk}), \dots,$

$\tilde{y}(x_{(j+1)k+s-1}))^T$ and analogically the vector

\underline{z}_j and $\tilde{y}(\underline{z}_j)$ by

$$\underline{z}_j = (x_{jk+s}, \dots, x_{(j+1)k+s-1})^T$$

and

$$\tilde{y}(\underline{z}_j) = (\tilde{y}(x_{jk+s}), \dots, \tilde{y}(x_{(j+1)k+s-1}))^T$$

and using the vector-valued functions $f(\underline{z}_j, \tilde{y}(\underline{z}_j))$ and $f(\underline{z}_j, \tilde{y}(\underline{z}_j))$ at the

corresponding points. If B, D be any $k \times s$ matrices and C any $k \times k$ matrix of constants, then the system:

$$\tilde{y}(\underline{z}_j) = B \tilde{y}(\underline{z}_{j-1}) + hC f(\underline{z}_j, \tilde{y}(\underline{z}_j)) + hD f(\underline{z}_j, \tilde{y}(\underline{z}_{j-1})) \quad (3)$$

represents an algorithm of the so called GBOM method.

We define the local truncation error of the method by the k -dimensional vector

$$\underline{L}(y(x), h) = \begin{bmatrix} y(s+w_s h) \\ \vdots \\ y(x+w_{k-1} h) \\ y(x+(m+w_0)h) \\ \vdots \\ y(x+(m+w_{s-1})h) \end{bmatrix} - hC \begin{bmatrix} y'(x+w_s h) \\ \vdots \\ y'(x+w_{k-1} h) \\ y'(x+(m+w_0)h) \\ \vdots \\ y'(x+(m+w_{s-1})h) \end{bmatrix}$$

$$- B \begin{bmatrix} y(x+w_0 h) \\ \vdots \\ y(x+w_{s-1} h) \end{bmatrix} - hD \begin{bmatrix} y'(x+w_0 h) \\ \vdots \\ y'(x+w_{s-1} h) \end{bmatrix} \quad (4)$$

If the first $k-s$ components of the local truncation error is of order h^p while the remaining s component are of order h^{p+1} we can give a weak definition of the order of the method as the order p ($p \geq 1$) w.r.t.s if and only if the following $kp+s$ equations are satisfied.

$$\sum_{j=1}^s b_{ij} = 1 \quad \text{for } i=1, \dots, k \quad (5)$$

$$w_{s-1+i}^q - \sum_{j=1}^s b_{ij} w_{j-1}^q = q \left[\sum_{j=1}^{k-s} w_{s-1+j}^{q-1} + \right.$$

$$\left. \sum_{j=k-s+1}^k c_{ij} (m+w_{s-k-1+j})^{q-1} + \sum_{j=1}^s d_{ij} w_{j-1}^{q-1} \right]$$

for $i=1, \dots, k-s, \quad q=1, \dots, p-1 \quad (6)$

$$(m+w_{s-k-1+i})^q - \sum_{j=1}^s b_{ij} w_{j-1}^q$$

$$= q \left[\sum_{j=1}^{k-s} c_{ij} w_{s-1+j}^{q-1} + \sum_{j=k-s+1}^k c_{ij} (m+w_{s-k-1+j})^{q-1} \right.$$

$$\left. + \sum_{j=1}^s d_{ij} w_{j-1}^{q-1} \right], \text{ for } i=k-s+1, \dots, k,$$

$q=1, \dots, p \quad (7)$

A-Stability Condition for a GBOM Method :

Solving the equation $y' = \alpha y$, $\text{Re} \alpha < 0$ equation (3) can take the form

$$(1-zC)\tilde{y}(z_j) = (B+zD)\tilde{y}(x_j), \quad \text{where } z=h$$

i.e.

$$\tilde{y}(z_j) = (I-zC)^{-1}(B+zD)\tilde{y}(x_j) \quad (8)$$

If $Q(z) = \det(I-zC)$

The last s equations of this vector is

$$\begin{bmatrix} y_{ik+k} \\ \vdots \\ y_{jk+k+s-1} \end{bmatrix} = \frac{1}{Q(z)} \begin{bmatrix} 0_{11}(z) \dots 0_{1s}(z) \\ \vdots \\ 0_{s1}(z) \dots 0_{ss}(z) \end{bmatrix} \begin{bmatrix} y_{ik} \\ \vdots \\ y_{ik+s-1} \end{bmatrix} \quad (9)$$

Or the form

$$Y_{n+1} = W(z) Y_n \quad (10)$$

Using the fixed-point theorem, it is clear that the necessary and sufficient condition for the A-stability of GBOM method (3) is

the necessary and sufficient condition for the convergence of the solution of the iterative system of equations (10) i.e.

$$||w(z)|| = \frac{1}{|Q(z)|} \left\| \begin{bmatrix} 0_{11}(z) \dots 0_{1s}(z) \\ \vdots \\ 0_{s1}(z) \dots 0_{ss}(z) \end{bmatrix} \right\| < 1$$

for $\text{Re } z < 0 \quad (11)$

Example for a GBOM Method :

Taking $k=2, s=1, n=1, w_0=0, w_1=\frac{1}{2}$ and solving equations (5), (6) and (7) as 9-equations in 8-unknown parameters i.e. the method is of order 4 and it can take the form

$$y_{n+1} = y_n + \frac{h}{24} (5f_n + 8f_{n+1} - f_{n+2}) \quad (12)$$

$$y_{n+2} = y_n + \frac{h}{6} (f_n + 4f_{n+1} + f_{n+2})$$

The A-Stability of the Method :

$$(I-zC) \begin{bmatrix} y_{2j+1} \\ y_{2j+2} \end{bmatrix} = (B+zD) y_{2j} \quad (13)$$

$$\begin{bmatrix} y_{2j+1} \\ y_{2j+2} \end{bmatrix} = \frac{1}{z^2-6z+12} \begin{bmatrix} 12 - \frac{z^2}{2} \\ z^2+6z+12 \end{bmatrix} y_{2j} \quad (14)$$

The last equation is

$$y_{2j+2} = \frac{z^2+6z+12}{z^2-6z+12} y_{2j} = w(z) y_{2j} \quad (15)$$

where $w(z)$ is the complex variables function

$$w(z) = \frac{z^2+6z+12}{z^2-6z+12} \quad \text{Re } z < 0 \quad (16)$$

Applying the maximum modulus principle. Then the modulus of this function attains its maximum value at the boundary

(i.e. at $z=iy, z=\infty$)

which gives $|w(z)| < 1$ from (10), (15) and (17) the methods (12) which is of order 4 is A-stable. (17)

References:

- 1) Dahlquist, G. "A Special Stability Problem for linear Multistep Method" *B I T 3* (1963) 27-43.
- 2) Gear, G.W. "Numerical Initial Value Problems in Ordinary Differential Equations" Printice-Hall, Inc. (1971) 212-222.
- 3) Hassan Nasr. "Generalized Blocking Over-implicit Multistep Method" *APLIKACE MATEMATIKY*, Under Printing (1979).
- 4) Hassan Nasr. "Necessary Conditions for the Generalized Blocking Over-implicit Multistep Method. " *APLIKACE MATEMATIKY* Under Printing (1979).

G.J. Cooper

University of Sussex

1. Introduction

Most methods used for the numerical solution of ordinary differential equations may be characterized by a pair of matrices (A, B) . Here, local error estimates for such methods are obtained as linear combinations of computed values. (This is equivalent to constructing pairs of imbedded methods.) The local error estimates depend on the assumption that exact solution values are used to start the step but it is shown that this assumption is not required for certain estimates. Some hybrid methods have such independent estimates, and also have simple step length change procedures.

Consider an initial value problem

$$x' = f(x), \quad x(t_0) = x_0, \quad f: \mathbb{R}^n \rightarrow \mathbb{R}^n,$$

and suppose that approximations are required on $[t_0, t_0 + H]$. Choose a step length h so that $Mh = H$, M a positive integer, and let $t_m = t_0 + mh$, $m = 1, 2, \dots, M$. A linear s -stage method computes, in step m ,

s n -element vectors,

$$y_i^{(m)} = \sum_{j=1}^s a_{ij} y_j^{(m-1)} + h \sum_{j=1}^s b_{ij} f(y_j^{(m)}),$$

for $i = 1, 2, \dots, s$, where $y_i^{(m)}$ approximates $x(t_{m-1} + c_i h)$ and $c = (c_1, c_2, \dots, c_s)^T$ is a consistency vector. The method may be expressed as

$$Y^{(m)} = AY^{(m-1)} + hBF(Y^{(m)}), \quad (1)$$

where $Y^{(m)} = y_1^{(m)} \oplus y_2^{(m)} \oplus \dots \oplus y_s^{(m)}$ and $F(Y^{(m)}) = f(y_1^{(m)}) \oplus f(y_2^{(m)}) \oplus \dots \oplus f(y_s^{(m)})$ are elements of \mathbb{R}^{ns} . The linear operators A and B are defined by matrices A and B , the rows of these matrices determining linear combinations of the s components of $Y^{(m-1)}$ and $F(Y^{(m)})$.

If step m is started with exact solution values

$$z_i^{(m)} = \sum_{j=1}^s a_{ij} x(t_{m-2} + c_j h) + h \sum_{j=1}^s b_{ij} f(z_j^{(m)}),$$

for $i = 1, 2, \dots, s$, and this may be expressed as

$$Z^{(m)} = AX^{(m-1)} + hBF(Z^{(m)}). \quad (2)$$

The local error in step m is $X^{(m)} - Z^{(m)}$ but the essential local error is defined to be $A^w(X^{(m)} - Z^{(m)})$ where w is a given integer associated with the method. For a Runge-Kutta method

$$A = \begin{pmatrix} 0 & \dots & 0 & 1 \\ 0 & & 0 & 1 \\ \vdots & & & \vdots \\ 0 & \dots & 0 & 1 \end{pmatrix}$$

and $w = 1$ so that the essential local error is merely the error in the final stage (repeated s times). For predictor-corrector pairs using methods of equal order, $w = 0$, but because these are multi-step methods some components of $X^{(m)} - Z^{(m)}$ are zero.

The aim is to determine a matrix Q so that, for the corresponding linear operator, $QZ^{(m)}$ estimates $A^w(X^{(m)} - Z^{(m)})$. For a Runge-Kutta method the rows of Q will be identical. For predictor-corrector pairs some rows of Q will be zero. For some estimates $QZ^{(m)}$ may be replaced by $QY^{(m)}$. That is, some estimators Q are independent of the localizing assumption and $QY^{(m)}$ estimates $A^w(X^{(m)} - Z^{(m)})$.

2. Conditions for error estimates

Let $p = (p_1, p_2, \dots, p_s)^T$ be a given vector with positive integer elements and define

$$\|y\|_M = \max_i M^{p_i} \|y_i\|, \quad Y = y_1 \oplus y_2 \oplus \dots \oplus y_s,$$

where $\|y\|$ is a norm on \mathbb{R}^n . Suppose a

method is used to compute $Y^{(1)}, Y^{(2)}, \dots, Y^{(M)}$ (which depend on h and hence M). If (for suitable starting values) $\|X^{(m)} - Y^{(m)}\|_M \leq k$, $m = 1, 2, \dots, M$, for all sufficiently large M , $M \geq M'$, then

$$\|x(t_{m-1} + c_i h) - y_i^{(m)}\| \leq Kh^{p_i}, \quad i = 1, 2, \dots, s,$$

and the method is said to be order p convergent.

A method is order p consistent if for $M \geq M'$

$$\|X^{(m)} - Z^{(m)}\|_M \leq K,$$

$$\|\underline{A}^w(X^{(m)} - Z^{(m)})\|_M \leq \frac{K}{M}, \quad m = 1, 2, \dots, M,$$

for some integer $w \geq 0$. (A more general definition is possible [3].) A stable hybrid method is order p convergent if it is order p consistent. In the present general context, a hybrid method is defined by

$$p_i < \max_j p_j \Rightarrow Ae_i = 0, \quad i = 1, 2, \dots, s,$$

where e_1, e_2, \dots, e_s is the natural basis for R^s . Since Ae_i is column i of A , this means that stage i in step $m-1$ is used in step m only if this stage is of maximum order.

To estimate the essential local error, a matrix Q is required so that (for $M \geq M'$)

$$\|\underline{A}^w(X^{(m)} - Z^{(m)}) - QZ^{(m)}\|_M \leq \frac{K'}{M^2},$$

$m = 1, 2, \dots, M$. Since $\underline{A}^w(X^{(m)} - Z^{(m)}) - QZ^{(m)} = (\underline{Q} + \underline{A}^w)(X^{(m)} - Z^{(m)}) - QX^{(m)}$, sufficient

conditions for Q to be an estimator may be obtained directly from the algebraic conditions for order p consistency [2]. Let $e = (1, 1, \dots, 1)^T$ and let C be the diagonal matrix whose diagonal elements are the elements of the consistency vector.

Theorem 1 Suppose a stable hybrid (A, B) method is order p consistent for a given order vector p with $p_i < 2 \min_j p_j$, $i = 1, 2, \dots, s$. Then Q is an estimator if, for each $i = 1, 2, \dots, s$,

$$e_i^T Q C^\tau e = 0, \quad \tau = 0, 1, \dots, p_i + 1,$$

$$e_i^T (Q + \underline{A}^w) BC^{\tau_0-1} \dots BC^{\tau_1-1}$$

$$[C^{\tau_0} - A(C-I)^{\tau_0} - BC^{\tau_0-1}]e = 0,$$

$\tau_0 + \tau_1 + \dots + \tau_\mu \leq p_i + 1$, where $\mu = 0, 1, 2, \dots$, and $\tau_0, \tau_1, \dots, \tau_\mu$ take all possible positive integer values.

3. Independent estimates

An estimate is independent of the localizing assumption if

$$\|Q(Z^{(m)} - Y^{(m)})\|_M \leq \frac{k'}{M^2}, \quad m = 1, 2, \dots, M.$$

Let $U^{(m)} = Z^{(m)} - Y^{(m)}$. Then (1) and (2) give

$$U^{(m)} = \underline{A}^m U^{(0)} + \sum_{i=1}^{m-1} \underline{A}^i (X^{(m-i)} - Z^{(m-i)}) + h \sum_{i=0}^{m-1} \underline{A}^i B [F(Z^{(m-i)}) - F(Y^{(m-i)})].$$

Let $m \geq v$, a positive integer, and assume $QA^v = 0$ so that the expression for $QU^{(m)}$ simplifies. The function differences may be replaced by a derivative expression and (1) applied again. With further assumptions it becomes possible to give conditions for independence. The following result is a particular case.

Theorem 2 Suppose the conditions of theorem 1 hold. Suppose that $QA = 0$ and $QBA^v = 0$, v a positive integer. Let $p_i = \tilde{p}$, $i = 1, 2, \dots, s$, and

$$QBA^T BC^{\tau_\mu-1} \dots BC^{\tau_1-1}$$

$$[C^{\tau_0} - A(C-I)^{\tau_0} - \tau_0 BC^{\tau_0-1}]e = 0,$$

$$\tau_0 + \tau_1 + \dots + \tau_\mu \leq \tilde{p},$$

where $\tau = 1, 2, 3, \dots$, $\mu = 0, 1, 2, \dots$ and $\tau_0, \tau_1, \dots, \tau_\mu$ take all positive integer values. Then (for suitable starting values)

$$\|Q(Z^{(m)} - Y^{(m)})\|_M \leq \frac{k'}{M^2}, \quad m \geq v, \quad M \geq M'.$$

Such an estimator is called a $Q(v)$ estimator. The condition $m \geq v$ is crucial because the estimate is independent of the localizing assumption only after v steps have been completed (and this applies to each step length change). The new starting conditions have to be chosen to maintain order p convergence.

Consider the Adams-Bashforth and Adams-Moulton methods of order k used in the PECE mode. A $Q(k)$ estimator may be obtained and this is just the Milne estimate. An extension of theorem 1 provides estimates for Runge-Kutta methods of order four (including the classical fourth order method). These estimates are obtained by imbedding, using three extra stages

(two extra function evaluations), but may not be independent.

The following method, represented as $p \mid A \mid B \mid Q \mid c$, is a hybrid method with a $Q(1)$ estimator (and $w = 1$). The first three stages give a hybrid method requiring two evaluations of f in each step. Since the second stage is of order three, and is not

used in the next step, it may be used to halve the step length. The final two stages are needed only for error estimation and one function evaluation is required in the next step; overall one extra evaluation of f is required. A similar method of order four is known.

3	0	0	1	0	0	0	0	0	0	0	0	0	-1	0	1	0
3	$\frac{1}{4}$	0	$\frac{3}{4}$	0	0	$\frac{3}{4}$	0	0	0	0	0	0	$-\frac{3}{4}$	0	$\frac{3}{4}$	$\frac{1}{2}$
3	$-\frac{1}{7}$	0	$\frac{8}{7}$	0	0	$-\frac{2}{7}$	$\frac{8}{7}$	0	0	0	0	0	$-\frac{8}{7}$	0	$\frac{8}{7}$	1
3	$\frac{5}{21}$	0	$\frac{16}{21}$	0	0	$\frac{452}{567}$	$-\frac{184}{567}$	$\frac{8}{81}$	0	0	0	0	$-\frac{16}{21}$	0	$\frac{16}{21}$	$\frac{1}{3}$
3	$-\frac{1}{7}$	0	$\frac{8}{7}$	0	0	$-\frac{8}{7}$	$-\frac{16}{7}$	$\frac{3}{7}$	$\frac{27}{7}$	0	0	0	$-\frac{8}{7}$	0	$\frac{8}{7}$	1

The method is also given by the equations

$$y_1^{(m)} = y_3^{(m-1)},$$

$$y_2^{(m)} = \frac{1}{4} y_1^{(m-1)} + \frac{3}{4} y_3^{(m-1)} + \frac{3}{4} hf(y_1^{(m)}),$$

$$y_3^{(m)} = -\frac{1}{7} y_1^{(m-1)} + \frac{8}{7} y_3^{(m-1)} - \frac{2}{7} hf(y_1^{(m)}) + \frac{8}{7} hf(y_2^{(m)}),$$

$$y_4^{(m)} = \frac{5}{21} y_1^{(m-1)} + \frac{16}{21} y_3^{(m-1)} + \frac{452}{567} hf(y_1^{(m)}) - \frac{184}{567} hf(y_2^{(m)}) + \frac{8}{81} hf(y_3^{(m)}),$$

$$y_5^{(m)} = -\frac{1}{7} y_1^{(m-1)} + \frac{8}{7} y_3^{(m-1)} - \frac{8}{7} hf(y_1^{(m)}) - \frac{16}{7} hf(y_2^{(m)}) + \frac{3}{7} hf(y_3^{(m)}) + \frac{27}{7} hf(y_4^{(m)}).$$

Since $z_1^{(m)} = x(t_{m-1})$ each component of the essential local error is proportional to $x(t_m) - z_3^{(m)}$ and the estimator gives just one estimate

$$x(t_m) - z_3^{(m)} = y_5^{(m)} - y_3^{(m)} + O(h^5).$$

- [1] J.C. Butcher, On the convergence of numerical solutions to ordinary differential equations, Math. Comput., 20 (1966), 1-10.
- [2] G.J. Cooper, The order of convergence of general linear methods for ordinary differential equations, SIAM J. Numer. Anal., 15 (1978), 643-61.
- [3] R.D. Skeel, Analysis of fixed step-size methods, SIAM J. Numer. Anal. 13 (1976), 664-685.

IN SEARCH OF A ROBUST INTEGRATION ALGORITHM
FOR GENERAL LIBRARY USE:
SOME TESTS, RESULTS AND RECOMMENDATIONS

M.B. Carver
Atomic Energy of Canada Limited
Chalk River Nuclear Laboratories
Chalk River, Ontario K0J 1J0

ABSTRACT

The design or selection of an integration algorithm suitable for general use in a subroutine library or within a simulation package is a demanding process, particularly as it is unrealistic to support a number of routines which purport to perform the same task. In addition to satisfying the criteria of excellence of documentation, reliability and efficiency which a numerical analyst would require of an integrator to be used in a specific project, a routine destined for general use must be exceptionally robust, as the average user has neither the desire, nor the specialized knowledge to interact effectively with the algorithm.

This paper examines the degree of robustness of selected integration algorithms by exposing them to a test profile of diverse problems. A number of common areas of difficulty are identified, and methods used to promote self sufficiency in these areas in the author's sparse matrix integration package GEARZ, are discussed.

SUMMARY

An extremely large number of algorithms have been proposed for the numerical solution of the ordinary differential equation initial value problem. The design or selection of such algorithms for use in a library of mathematical subroutines for general scientific and engineering use is difficult, as it is impractical to maintain a large number of subroutines which attempt to do the same job with varying degrees of success.

Criteria for selection must include accuracy, efficiency and ease of use, thus one can consider realistically only those algorithms which have options to determine an optimal integration step size, and possibly order, by means of a built-in estimate of the associated truncation error, and are presented as quality software, complete with detailed internal and external documentation which emphasizes the weaknesses as well as the strengths of the method. These restrictions considerably reduce the possibilities, but a number of candidates remain. Typical amongst these are algorithms published by Gear[1], Hindmarsh[2], Byrne[3], Hamming[4] and Krogh[5], all of which satisfy the above criteria.

Because these algorithms are reliable, quality software, numerical analysts have incorporated them in a multitude of applications packages, in which the user is shielded from the complexities involved in management of the integration. This type of application has in fact been their greatest success. To the uninitiated user of a subroutine library, however, their correct implementation can be seen as a prohibitive task.

In order to focus further on quality for general use, the criterion of robustness must be added, a robust algorithm being defined as one which produces the result to the desired accuracy or a clear indication of failure, requires a minimum of effort and does not demand clairvoyance from the user. This criterion is essential because the majority of projected users are not numerical differential equation experts, and therefore, not qualified to make decisions on fundamental issues such as the initial step size, error base, sparsity, and error recovery. Unfortunately, as such decisions are frequently left to the user's discretion in the guise of generality, the application of this robustness criterion eliminates most of the above-mentioned candidate algorithms.

This paper attempts to show that such algorithms can be made considerably more robust at the expense of a negligible loss of generality by further automation and simplification of the decision process during start up, integration, error processing, and discontinuity handling. Several specific areas where traditionally required user interaction can be either eliminated or reduced to optional status to improve effectiveness are discussed. They are:

- (a) Initial Step Size Selection: A routine with efficient step size adjustment is far better qualified than a user to provide this selection.
- (b) Tolerable Error Imposition: An algorithm normally accepts a step if the estimated local truncation error in Y_j satisfies $\epsilon_j < \text{TOL} * Y_{\text{base}_j}$, where Y_{base_j} is frequently defined as the maximum attained value of Y_j , or made available as an array for user specification. Again users have no criteria on which to nominate each Y_{base_j} , but automatic

function $t_{Ybase} = \max(|Y|, Y_{sig})$ is general, and requires the user only to provide Y_{sig} , the value below which any $|Y|$ is effectively zero, if this differs from the machine unit round off.

(c) **Jacobian Evaluation and Handling:** Most general integrators require estimates of the Jacobian matrix from time to time, and the user is often left to speculate on a choice of how these should be obtained. Sophisticated users may gain efficiency by coding their own Jacobian, but within a general package this possibility grows more remote, and could end up less efficient than numerical evaluation. Standard numerical procedure is to perturb each individual Y_i to obtain the set $\partial F_i / \partial Y_j$, requiring n function evaluation calls. This may be optimized; for example, a tridiagonal matrix Jacobian may be evaluated using only three calls, and proportionate savings may be realized in sparse matrices[6]. Sparse matrix Jacobian evaluation and handling optimized in this way is invaluable for general PDE/ODE packages in which structure is arbitrary.

(d) **Discontinuity Detection and Negotiation:** A switch in equation definition occurring at an unknown time dependent on critical relationships evolving during integration is a common physical phenomenon, but is traumatic for an integrator and frequently leads to time limit or termination. A robust integrator should detect such a discontinuity and transcend it, issuing a suitable terse report. Optional user interaction to facilitate smoother handling of discontinuities should be made available, for example see reference [7].

(e) **Stiffness Detection:** The most common way a standard non-stiff integration algorithm detects stiffness is to run to time limit. This is unsatisfactory as the results produced will be of little value. A far less expensive approach is to report suspected stiffness[4] and either refuse to waste further time computing, or if possible, switch to a stiff algorithm.

(f) **Calling Sequence and Common Blocks:** In a package, the user is shielded from direct communication through the integrator parameter list, but this is mentioned here to facilitate library or package interface design. Any integrator calling sequence requires ten parameters, the derivative subroutine, dependent variable and derivative vectors and their size, independent variable and interval, tolerance, warning flag, one working storage array and current step size. Only these parameters are mandatory and all other communication may be through optional common blocks. The integrator itself can assign parts of the working storage array in predictable sequence, and choose a suitable method for handling the Jacobian from an assessment of the equation structure and available working storage.

A number of results of both academic and applications problems are given to illustrate that the modifications proposed above are not only more robust, but frequently more efficient.

Finally, a brief description is given of the GEARZ package, which incorporates the modifications discussed into a combination of the Hindmarsh-Gear algorithms[2] with the Curtis-Reid sparse matrix routines[6], and is currently used on the Atomic Energy of Canada Limited subroutine library and the FORSIM[8], and MACKSIM[9] simulation packages.

REFERENCES

1. C.W. Gear, The Automatic Integration of Ordinary Differential Equations, Comm. ACM, V14, 3, p.176, March 1971.
2. A.C. Hindmarsh, The LLL Family of Ordinary Differential Equation Solvers, UCRL-78129, April 1976.
3. G. Byrne and A.C. Hindmarsh, A Polyalgorithm for the Numerical Solution of Ordinary Differential Equations, ACM TOMS, V1, p.71, 1975.
4. L.F. Shampine, H.W. Watts and S.M. Davenport, Solving Non-stiff Ordinary Differential Equations - The State of the Art, Siam Review, V18, 3, 1976.
5. F.T. Krogh, Variable Order Integrators for the Numerical Solution of Ordinary Differential Equations, Jet Propulsion Lab., Section 314 Subroutine Write-up, May 1969.
6. A.R. Curtis and J.K. Reid, FORTRAN Subroutines for the Solution of Sparse Sets of Linear Equations, AERE-R-6844, 1971.
7. M.B. Carver and S.R. MacEwen, Analysis of a System Described by Implicitly Defined Ordinary Differential Equations Containing Discontinuities, Applied Mathematical Modelling, V2, 1978, p.280.
8. M.B. Carver, D.G. Stewart, J.M. Blair and W.N. Selander, The FORSIM VI Package for Automated Solution of Arbitrarily Defined PDE/ODE Systems, Atomic Energy of Canada Limited report AECL-5821, February 1978.
9. M.B. Carver and A.W. Boyd, A Program Package Using Stiff Sparse Techniques for the Automatic Solution of Mass Action Chemical Kinetics, submitted to Int. J. Chem. Kinet.
10. M.B. Carver and A.P. Baudouin, Solution of Reactor Kinetics Problems Using Sparse Matrix Techniques in an ODE Integrator, Atomic Energy of Canada Limited report AECL-5177, 1975.

ON AN ITERATIVE IMPROVEMENT OF THE APPROXIMATE
SOLUTION OF SOME ORDINARY DIFFERENTIAL EQUATIONS

P. E. Zadunaisky

(CNE-Observatorio Nacional de Física Cósmica, San Miguel, Argentina)

G. Lafferriere

(Facultad de Ciencias Exactas, Universidad de Buenos Aires, Argentina)

ABSTRACT

Let us consider a system of ordinary differential equations of the form $f(x, y, y', \dots, y^{(m)}) = 0$ where y and f are vector functions of x . By introducing an operator T such that $Tu = f(x, u, u', \dots, u^{(m)})$ we have $Ty = 0$. Assuming that y^0 is an approximation of the solution $y(x)$ a generalisation of Newton's method can be applied to improve, under certain conditions, such approximation by the recursive algorithm $y^{i+1} = y^i - T'(y^i)^{-1} \cdot Ty^i$ ($i = 0, 1, 2, \dots$) and assuming that $T'(y^i)^{-1}$ exists.

In the present case we can use such an approach in a numerical fashion as follows. After obtaining by any method of integration numerical approximations y_n on a discrete set of points x_n ($n = 1, \dots, N$) we interpolate them by a convenient function $P(x)$. By taking this interpolant as the first analytical approximation y^0 Newton's process is applied pointwise in order to correct by iterations the discrete approximations y_n . This procedure may become rapidly convergent especially in some stiff problems where we have obtained so far promising results.

A Portable Automatic Taylor Series (ATS) Compiler
for Solving Ordinary Differential Equations*

by

Y.F. Chang, Roy Morris⁺, Computer Science Department, University of Nebraska
Lincoln, Nebraska 68588

and

George Corliss, Department of Mathematics and Statistics, Marquette University
Milwaukee, Wisconsin 53233

EXTENDED ABSTRACT

A portable compiler computer program is developed for the automatic numerical solutions of ordinary differential equations. The inputs to this compiler are FORTRAN statements of the equations (differential and algebraic) and of the initial conditions. The output of this compiler is an object program written in FORTRAN. The Automatic Taylor Series (ATS) method of solving ordinary differential equations is based on the premise that differentiation of functions can be performed at specified points by simple additions and multiplications. The solution of very complicated ordinary differential equation can be expanded into a long Taylor series with ease. The availability of very long Taylor series has led to new understanding of the convergence property of the series in relation to the locations of singularities and to the radius of convergence. It is then possible to set an error limit beforehand and maintain local error control over the computations within that limit up to almost machine accuracy. The power of the Automatic Taylor Series (ATS) method lies in the speed and accuracy that is obtainable.

The foundation of the Automatic Taylor Series (ATS) method is the replacement of differentiation of functions by simple arithmetic operations. The first publication on this simplification was by Wilson[1] in 1949. Gibbons[2] obtained the recursive relations for the fully automatic differentiation of some simple functions in 1960. A summary of power series method was published by Leavitt[3] in 1966. Application of power series to the automatic solutions of ordinary differential equations was first discussed by Moore[4] in 1966. Barton, et al[5] in 1970 first reported implementation of an automatic solution. That automatic solution, by Barton, et al, was obtained through a compiler program that was written in a machine-dependent language. One of the present authors published an account of a compiler program written in FORTRAN for the automatic solutions of ordinary differential equations[6]

in 1974. The principal differences between this and the one by Barton, et al are the calculation for the series' radius of convergence and the concomitant error control. This compiler was written in FORTRAN with hope that it would be portable. This proved to be optimistic, not factual. There are, at present, two versions of this compiler: one for the CDC-computer, and one for the IBM-computer. These two versions are quite distinct.

The present paper is the report of a portable compiler program for the automatic solutions of ordinary differential equations. This program is written in FORTRAN with special treatment of machine-dependent parameters to obtain portability. Parameters such as bits/character, character/word, single-precision limit, and double-precision limit are stored in a single Blockdata location for easy access and quick change. Since the bits/character and word length are machine-dependent, we have also instituted a numerical code for character strings. This allows for easy character string manipulations using integer arithmetic; thus, we have overcome the difficulty of character string manipulation on certain computers. As development proceeds, testing is being done on computers from several manufacturers: IBM, CDC, PDP, and Xerox Sigma.

The input to this compiler program consists of four blocks of user specified information: the statement of differential equations and three blocks of instructions to be inserted into the object program. Each of the data blocks ends with a data terminator (\$), and only one data terminator may appear on a line. The differential equations are entered in the first block, with (DIFF) as the differential operator. For example, DIFF(Y,X,2) denotes $y''(x)$. The rest of the expressions are entered in typical FORTRAN statements.

For example, to solve the problem:

$$y'' = y' \sin(y) \exp(x);$$

$$y(0) = 1, \quad y'(0) = -1 \text{ on } [0,4]$$

the input would be

* This work is in part supported by N.S.F. Grant #MCS78-03022.

⁺ Present address: Department of Computer Science, Iowa State University, Ames, Iowa 50011

Column 7

```
|
DIFF(Y,X,2) = DY*SIN(Y)*EXP(X)
DY = DIFF (Y,X,1)  $
$
START = 0.0
END = 4.0
Y (1) = 1.0
Y (2) = - 1.0  $
$
```

This is an example of the simplest possible input. The use of the first and third data blocks is mandatory, while the use of the second, and fourth data blocks is optional. The sophisticated user may control the solution of very complicated problems through these optional data blocks.

The second data block may be used to enter non-executable FORTRAN statements, such as COMMON, DATA, or SUBROUTINE, into the object program at the appropriate location. Using the SUBROUTINE statement allows the user to interlace many complicated programs together. The third data block is used to define the solution interval and initial conditions. The fourth data block may be used for such things as changing the amount of printout, starting and stopping the solution at any given point in either the dependent or independent variables, and the calling of other subroutines. Problems with piecewise analytic solutions can be solved with ease.

This compiler program reads the input and generates a FORTRAN object program, which subsequently is compiled and run to solve the specified problem. The object FORTRAN program generated by this compiler conforms as closely as is possible to the general structure suggested by Hull and Enright[7]. As the first data block of equations are read in, this compiler produces a list of tokens. Each token represents a constituent in the equations. Variables and constants are stored in arrays as integers with the corresponding tokens pointing to the proper element in the arrays. The token produced for a function is an integer code representing that function.

The list of tokens are checked for grammatical correctness and put in Polish postfix form. The grammar used is an LL(1) grammar, which has the property that the top stack symbol and the input token uniquely determine which production of the grammar should be applied next. The postfix expression is then arranged in triplets of operator-operand-operand, from which the final object program is generated.

This compiler is a significant improvement over the earlier compilers for the automatic solution of ordinary differential equations. Not only is it portable between different computers, by the simple change of an easily accessible blockdata, there are other important features. The calculation for the series' radius of convergence has been improved such that the order of the singularity in the complex plane of the solution function can be estimated with a certain degree of accuracy. We have also included in this compiler the ability to vary the ratio of step-size to the radius of convergence and variable order in the numerical integration. Other

improvement include the choice of single- or double-precision, and comprehensive diagnostic error messages.

Copies of this compiler will be made available to researchers in the field of numerical ordinary differential equations upon request, when all of the major bugs have been removed and full documentation is ready.

REFERENCES

- (1) E.M. Wilson, A Note on the Numerical Integration of Differential Equations, Quar. J. Mech. & Appl. Math., Vol. 2 (1949), p. 208.
- (2) A. Gibbons, A Program for the Automatic Integration of Differential Equations Using the Method of Taylor Series, Comp. J., Vol. 3 (1960), p. 108.
- (3) J.A. Leavitt, Methods and Applications of Power Series, Math. Comp., Vol. 20 (1969), p. 46.
- (4) R.E. Moore, Interval Analysis, Prentice-Hall, Englewood Cliffs, New Jersey, 1966.
- (5) D. Barton, I.M. Willers, and R.V.M. Zahar, The Automatic Solution of Systems of Ordinary Differential Equations by the Method of Taylor Series, Comp. J., Vol. 14 (1970), p. 243.
- (6) Y.F. Chang, Automatic Solution of Differential Equations, Lecture Notes in Math. #430, Constructive & Computational Methods for Differential and Integral Equations, D.L. Colton and R.P. Gilbert, ed., Springer-Verlag, New York, 1974, p. 61.
- (7) T.E. Hull and W.H. Enright, A Structure for Programs that Solve Ordinary Differential Equations, Dept. of Computer Science Technical Report #66, Univ. of Toronto, Toronto, Canada (1974).

ROSENBROCK-TYPE METHODS FOR THE NUMERICAL SOLUTION OF STIFF SYSTEMS

Peter Kaps

University of Innsbruck

Abstract: Generalized Runge-Kutta methods for the numerical integration of stiff systems are investigated. A simple way of obtaining the order conditions is given. Numerical results of two embedded (3)4 methods (GRK4A, A-stable, and GRK4T, A(89⁰)-stable with small truncation errors) are presented. Comparing computing time both methods are competitive with the GEAR program.

In this paper I want to give a survey of the results of my thesis [1] and two articles I wrote together with P. Rentrop [2] and G. Wanner [3].

For the numerical solution of the initial value problem

$$y'(x) = f(y(x)), \quad y(x_0) = y_0$$

defined in an n-dimensional real or complex space the following method is considered:

$$E = I - \gamma h f'(y_0)$$

$$Ek_1 = hf(y_0)$$

$$Ek_2 = hf(y_0 + \alpha_{21}k_1) + hf'(y_0)\gamma_{21}k_1$$

⋮
⋮
⋮

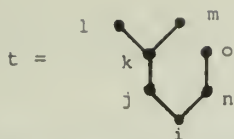
$$Ek_s = hf(y_0 + \alpha_{s1}k_1 + \dots + \alpha_{s,s-1}k_{s-1}) + hf'(y_0)(\gamma_{s1}k_1 + \dots + \gamma_{s,s-1}k_{s-1})$$

$$y_1 = y_0 + \sum_{i=1}^s c_i k_i.$$

Here γ , α_{ij} , γ_{ij} , c_i are real coefficients, h denotes the step size, $f'(y_0)$ the Jacobian, I the identity matrix and s the number of stages. The method needs one evaluation of f' , the solution of one linear system for s right-hand sides and up to s evaluations of f . The matrix-vector multiplications can be avoided by a suitable transformation.

Similar methods were first investigated by H.H. Rosenbrock [4]. G. Wanner introduced the coefficients γ_{ij} . So we call these methods Rosenbrock-Wanner or short ROW methods.

The equations of condition have a structure similar to Runge-Kutta methods. In [1], [3] they are derived with the help of Butcher series. In order to illustrate how easy they can be obtained, we give as an example the equation for the tree



The tree t has 7 nodes and corresponds to an elementary differential of order 7. Everybody who has ever tried to obtain the equations of condition by Taylor series expansion will appreciate the following simple procedure: Attach to each node of t a summation index i, j, k, \dots . Put $\beta'_{ij} = \alpha_{ij} + \gamma_{ij} + \gamma\delta_{ij}$. Then we must have

$$7.4.3.2 \quad \sum_{i,j,k,l,m,n,o} c_i \alpha_{ij} \beta'_{jk} \alpha_{kl} \alpha_{km} \alpha_{in} \beta'_{no} = 1.$$

The product of integers before the sum is Butcher's coefficient $\gamma(t)$, but let us call it $\Gamma(t)$ here in order to avoid any confusion with the coefficient γ of the ROW method. Then there is a sum over all summation indices attached to the nodes of t . The summands are products containing the factors c_i for the root of the tree, α_{pq} (or β'_{pq}) whenever a multiply (or singly) branched node p is connected directly upwards with a node q .

The matrix (α_{ij}) has elements $\neq 0$ below the diagonal only, but (β'_{ij}) has non-zero diagonal elements. Putting $\beta_{ij} = \beta'_{ij} - \gamma\delta_{ij}$ one obtains

$$\sum_{i,j,k,l,m,n,o} c_i \alpha_{ij} \beta_{jk} \alpha_{kl} \alpha_{km} \alpha_{in} \beta_{no} = \frac{1}{\Gamma(t)} - \gamma \left(\frac{1}{\Gamma(t_1)} + \frac{1}{\Gamma(t_2)} \right) + \gamma^2 \frac{1}{\Gamma(t_3)},$$

where $t_1 =$ $,$ $t_2 =$ $,$ and $t_3 =$

are the trees which result from t by omitting 1 or 2 singly branched nodes. By this procedure the number of summands is considerably reduced. The equations of condition obtained in this way are listed in [1], [2], [3] up to order 5.

ROW methods have good stability properties. The stability function is a rational approximation to the exponential function with denominator $(1-\gamma z)^S$. If the number of stages does not exceed the order the numerator P is given by

$$P = \sum_{k=0}^S z^k \sum_{i=0}^k \binom{S}{i} \frac{(-\gamma)^i}{(k-i)!}.$$

The coefficient of z^k is a generalized Laguerre polynomial in $\frac{1}{\gamma}$.

Unfortunately, one can not obtain γ -values leading not only to small truncation errors but also to A-stability. Therefore, we worked out $A(\alpha)$ -stable methods with small truncation errors and A-(or L-)stable methods.

Reasonable integration methods need also a step size control. We investigated in [1], [3] methods using passive Richardson extrapolation (that is two steps with step size h and one with $2h$ for error estimation). The following table gives the minimal number of stages s_m necessary for a method of order p :

p	4	5	6
s_m	3	5	6

In [2] embedded methods of orders (3)4 were studied. The solution of order 4 was used for step continuation. Four stages were necessary, but only 3 function evaluations because

$$\alpha_{31} = \alpha_{41}, \alpha_{32} = \alpha_{42}, \alpha_{43} = 0.$$

Both types of methods work well. As an example the embedded methods GRK4A (A-stable) and GRK4T (small truncation error, A(89.3°)-stable) are compared with GEAR for the 25 test examples of Enright, Hull, Lindberg [5]. The Jacobian was calculated by difference approximation which needs n function evaluations. For step size control the formula $h_{\text{new}} = 0.9 h_{\text{old}} (\text{TOL}/\text{EST})^{1/4}$, $0.5 h_{\text{old}} \leq h_{\text{new}} < 1.5 h_{\text{old}}$ was used. $\text{EST} = \|\hat{y}_{1\text{old}} - y_{1\text{old}}\|$ is the estimate (\hat{y}_1 and y_1 denote the solutions of orders 3 and 4 respectively and $\|\cdot\|$ the maximum norm) and TOL is the given tolerance. The safety factor 0.9 and the limitation of h_{new} are fixed by experience. A new step is accepted if the error does not exceed the tolerance. For components smaller than 1 the absolute error was used, and for the others the relative error. In the following table the computing time TZ in seconds and the total number of function evaluations TF are listed for $\text{TOL} = 10^{-4}$ and initial step size 10^{-3} .

	All examples		All examples except B5, E4	
	TZ	TF	TZ	TF
GRK4A	18.85	14428	16.02	12431
GRK4T	19.23	14181	14.21	11043
GEAR	44.21	9099	21.57	5423

(The computations were performed in FORTRAN single precision with a 38 bit mantissa (11 decimals) on the TR440 of the Leibniz-Rechenzentrum of the Bavarian Academy of Science). GEAR has difficulties in B5, where computing time reaches 21.23 seconds. In E4, GRK4T is approximately 4 times worse than GRK4A. This is due to a positive eigenvalue in E4 and disappears for an exact evaluation of the Jacobian.

Acknowledgement

The author is indebted to E. Hairer for many useful suggestions.

References

- [1] P. KAPS: Modifizierte Rosenbrockmethoden der Ordnung 4,5, und 6 zur numerischen Integration steifer Differentialgleichungen, Dissertation, Innsbruck 1977.
- [2] P. KAPS, P. RENTROP: Generalized Runge-Kutta Methods of Order Four with Step Size Control for Stiff Ordinary Differential Equations, appears in Numer.Math.
- [3] P. KAPS, G. WANNER: Rosenbrock-Type Methods of High Order, in preparation.
- [4] H.H. ROSENBRICK: Some general implicit processes for the numerical solution of differential equations, Comp.J. 5, 1963, 329-331.
- [5] W.H. ENRIGHT, T.E. HULL, B. LINDBERG: Comparing numerical methods for stiff systems of ordinary differential equations, Tech.Rep. No 69, 1974, University of Toronto, see also BIT 15, 1975, 10-48.

Author's address: P. Kaps, Mathematik I, Technikerstr.13, A-6020 Innsbruck, Austria, Europe

DETERMINING REALISTIC INITIAL VALUES FOR
STIFF SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS
OCCURRING IN IONOSPHERIC PHYSICS

David S. Watkins
Department of Mathematics
Utah State University
Logan, Utah 84322

We consider an initial value problem for a stiff system of first order differential equations which occurs in the modeling of flows of ions in the upper ionosphere. The problem is that we know the values for some, but not all, of the initial conditions. The dependent variables for which we are solving are electron temperature, heat flow, and stress. In this context stress means temperature anisotropy, i.e. the difference between temperature parallel and perpendicular to the earth's magnetic field. Realistic initial values for temperature and heat flow are known from satellite measurements. The stress value is not known because satellite measurements have generally assumed an isotropic temperature distribution. We shall see that we can exploit the stiffness of the system to calculate a realistic stress value.

The choice of initial values is critical to the nature of the solution, particularly in the early stages. Most choices of initial conditions give rise to solutions which have a rapid, short-term fluctuation which soon dies out. This behavior is typical of stiff systems, but not of the physical system under consideration. For a typical (arbitrary) choice of initial stresses, temperature fluctuations on the order of 100 K in one kilometer are observed in the early stages of the numerical solution. (Note: the physical problem is an initial value problem in the mathematical sense only. The independent variable is altitude, not time. We start at a convenient initial altitude and march outward from the earth a distance of some ten thousand kilometers.) Satellite measurements indicate that such a fluctuation is physically unrealistic. Temperature changes of this magnitude actually take place over tens or hundreds of kilometers. The short term fluctuations in the numerical solution are caused by the wrong choice of initial stress. Therefore, we wish to choose our unknown initial stress in such a way as to obtain a solution in which rapid, short-term fluctuations are minimized or eliminated. Such a solution is said to be in stiff equilibrium. A general procedure for determining initial conditions for which the solution is in stiff equilibrium is developed and justified here. We consider the linear problem first and show how the unknown initial values can be calculated by a direct procedure. Then we show how the non-linear problem can be solved by a procedure which iterates the direct procedure of the linear problem. Finally, we give numerical

results showing that the procedure does work for the problem under consideration.

MATHEMATICAL DISCUSSION

We consider first the linear problem $y' = Ay + b$. We assume that the system is stiff, i.e. that one or more of the eigenvalues of A has large negative real part, and the other eigenvalues are comparatively small in absolute value. Let $\lambda_1, \lambda_2, \dots, \lambda_n$ be the eigenvalues, and assume that they are ordered so that $\lambda_1, \dots, \lambda_k$ are the "small" eigenvalues and $\lambda_{k+1}, \dots, \lambda_n$ are the "stiff" eigenvalues, i.e. those having large negative real part. Given any initial condition $y(x_0) = y_0$, the initial value problem has a unique solution. If A is simple the solution is

$$y = \sum_{j=1}^n v^{(j)} \hat{y}_j(x)$$

where

$$\hat{y}_j(x) = \begin{cases} c_j \exp(\lambda_j(x-x_0)) - \hat{b}_j \lambda_j^{-1} & \text{if } \lambda_j \neq 0 \\ c_j + \hat{b}_j(x-x_0) & \text{if } \lambda_j = 0. \end{cases}$$

$v^{(j)}$ is the eigenvector corresponding to λ_j , the \hat{b}_j are constants determined by b and A , and the c_j are constant determined by the initial vector y_0 .

In our problem we assume that some of the initial values (components of y_0) are known and others are unknown. We wish to specify these unknown values in such a way that rapid, short-term fluctuations in the solution are minimized. These short-term fluctuations are caused by those terms of the form $c_j \exp(\lambda_j(x-x_0))$ for which λ_j has large negative real part. We can eliminate these terms by specifying the initial values in such a way that $c_j = 0$, $j = k+1, \dots, n$. A solution having this property is said to be in stiff equilibrium. Not surprisingly, in order to obtain a solution which is in stiff equilibrium, the number of initial values which we allow to vary must be equal to be number of stiff eigenvalues. This restricts the class of problems which can be solved, but the restriction is not as severe as it might at first seem because there is usually some latitude in the choice of stiff eigenvalues. For example, if A has eigenvalues $1, 0, -1, -10^2, -10^3$,

-10^6 , we may choose to designate one, two, or three eigenvalues as stiff.

By a straightforward procedure we can determine the initial values which give a solution in stiff equilibrium. First, $\hat{y}_j(x_0) = c_j - \mu_j b_j$, $j=1, \dots, n$, where $\mu_j = \lambda_j^{-1}$ if $\lambda_j \neq 0$ and $\mu_j = 0$ if $\lambda_j = 0$. These equations can be expressed in matrix form as $\hat{y}(x_0) = c - Mb$, where M is the diagonal matrix whose j th main diagonal entry is μ_j . Thus $y_0 = V(c - Mb)$, where V is the matrix whose j th column is $v^{(j)}$. This matrix equation can be written in block form

$$(1) \begin{bmatrix} w_0 \\ z_0 \end{bmatrix} = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} \begin{bmatrix} d-r \\ e-s \end{bmatrix}$$

where z_0 consists of those components of y_0 which are unknown, e consists of those components of c which are to be set to zero, and r and s are components of the known vector Mb . For any specified w_0 we wish to find z_0 such that this system has a unique solution $c = \begin{bmatrix} d \\ e \end{bmatrix}$ with $e = 0$.

Such a z_0 must satisfy the equations

$$w_0 = V_{11}(d-r) - V_{12}s$$

$$z_0 = V_{21}(d-r) - V_{22}s$$

obtained from (1) by setting $e = 0$. These equations have a unique solution if and only if V_{11} is nonsingular. If V_{11} is nonsingular we can solve the first equation uniquely for $d - r$. We can then substitute $d - r$ into the second equation to calculate z_0 .

The procedure which we have just outlined includes the calculation of the eigenvalues and eigenvectors of A and the solution of two linear systems. This would be costly for a large system of differential equations, but for a system of modest size (one of ten equations, say) these operations are not expensive.

Now consider the nonlinear problem $y' = f(y)$, $y(x_0) = y_0$, where f is assumed twice continuously differentiable. We are concerned here with the behavior of the solution in the vicinity of y . Near y_0 the solution to the nonlinear problem is well approximated by the solution to the linear problem

$$y' = f(y_0) + \frac{\partial f}{\partial y}(y_0)(y - y_0), \quad y(x_0) = y_0.$$

This problem has the form $y' = Ay + b$, where

$$A = \frac{\partial f}{\partial y}(y_0) \text{ and } b = f(y_0) - \frac{\partial f}{\partial y}(y_0)y_0.$$

As in the linear case, y_0 contains some unknown components, so we cannot simply proceed as in the linear case. Instead we must use an iterative procedure. We make an initial guess of y_0 , use this initial guess to calculate A and b , and proceed as in the linear case to calculate a new guess of y_0 . We repeat this procedure until (hopefully) it con-

verges.

In the nonlinear case the partitioning of the eigenvalues is complicated by the fact that the matrix A changes from iteration to iteration, and the eigenvalues change with it. Thus, a partition which is reasonable on one iteration may not be reasonable on the next. It is impractical to change the number of initial values which are being adjusted from one iteration to the next, so we must decide at the beginning how many we wish to vary and stick with that number. If we decide to let i initial values vary, then at each iteration we place the i "stiffest" eigenvalues in the "stiff" set and all others in the "small" set. Once the iterations have converged to some y_0 , we can check whether the partition is reasonable by examining the eigenvalues of $\frac{\partial f}{\partial y}(y_0)$. We can also monitor the partition at intermediate iterations, but this is not necessary. All that really matters is whether the final partition is reasonable.

NUMERICAL RESULTS

We present here some numerical results for the system of transport equations discussed in the introduction. Consider the following initial conditions: altitude = 1500 km, temperature = 2000 K, heat flow = -1.943×10^{-5} erg(cm) $^{-2}$ (sec) $^{-1}$. The unknown initial condition is stress. The following table shows the stress values calculated by the method proposed here.

Iteration	Stress (K)
0	0.0
1	-5.159232
2	-5.163496
3	-5.163496

We see that the procedure converges to seven decimal places in two iterations. At each iteration the Jacobian matrix was estimated by numerical differencing. The eigenvalues and eigenvectors were calculated by the IMSL subroutine ELGRF, which uses a shifted QR algorithm.

The initial values have been partitioned into three known values (counting altitude, the independent variable) and one unknown value. In order for this to be a reasonable partition we must be able to partition the Jacobian into a set of three "small" eigenvalues and a set of one "stiff" eigenvalue. The following table shows the eigenvalues of the Jacobian computed on the final iteration.

eigenvalues	scale heights
1 7.18×10^{-10}	
2 0.0	
3 -4.47×10^{-9}	2.24×10^8 cm = 2240 km
4 -7.80×10^{-6}	1.28×10^5 cm = 1.28 km

If we designate the first three eigenvalues "small" and the fourth one "stiff", then the "stiff" eigenvalue is over 1000 times larger in absolute value than any of the "small" ones. Thus the partition is reasonable.

The fourth eigenvalue shows that the system

is quite stiff. It looks small but it is actually large, given the units of the problem. In the solution the term which corresponds to eigenvalue 1 decays by a factor e in the distance ~ 1 . This distance is commonly called the scale height of the term. We are using cgs units, so the scale height is in centimeters. If we express this distance in kilometers, then the scale height of the term corresponding to the fourth eigenvalue is 1.28 km. This is an extremely short distance, given that we are integrating out some ten thousand kilometers. By contrast, the scale height associated with the next largest eigenvalue is over 2000 km.

Finally, we have to ask whether the answer given by this procedure is meaningful. Is -5.163496 really the "right" initial stress value? Yes, it is. If we integrate outward using this initial value we get smooth profiles which do not show any short term fluctuations. The use of any other initial stress value leads to large initial fluctuations in the profiles which are unrealistic from a physical standpoint.

A COMPARISON OF METHODS
FOR COMPUTING STURM-
LIOUVILLE EIGENVALUES

J.W. PAINE
Australian National University

An important source of Initial value problems arises from the use of shooting methods to solve Sturm-Liouville eigenvalue problems of the form

$$(1.a) \quad -(p'u)' + qu = \lambda ru \quad x \in (a,b)$$

$$(1.b) \quad \alpha u(a) + \beta u'(a) = 0 \\ \gamma u(b) + \delta u'(b) = 0$$

$$(1.c) \quad u, p'u \in C[a,b]$$

Because of the large variety of methods available for approximating the eigenvalues of this problem, and because of their different underlying assumptions on which the methods are based, it is important to know and to understand the different properties of the eigenvalue approximations which will be obtained.

The most important property that any method should possess is convergence, and (if we assume that each method is parameterized over the set of positive integers) in it's simplest form this is expressed as

$$\lim_{N \rightarrow \infty} \lambda_{\kappa}^{(N)} = \lambda_{\kappa}$$

where λ_{κ} is the κ 'th exact eigenvalue and $\lambda_{\kappa}^{(N)}$ is the κ 'th eigenvalue of the N 'th approximating problem. But although such basic convergence results are necessary, they are not altogether useful since they do not really give much information about the approximations that will be obtained for a fixed finite value of N . As a result of this, the approximation results for most methods are refined, and given in the form

$$|\lambda_{\kappa} - \lambda_{\kappa}^{(N)}| \leq C N^{-p} \quad ; N > N_0(\kappa)$$

where C generally depends on κ but is independent of N .

For many situations such estimates are sufficient to guide one's choice of method. However since these results are local convergence results in the sense that for a fixed value of κ , we can find an N_0 large enough such that the convergence estimate is valid for every $N > N_0$, there are situations when the information given by such local convergence results will be

insufficient or even misleading, when comparing different methods. One such situation is when there is a large number of eigenvalues to be approximated and there is a definite practical need to keep the size of N as small as possible.

An example of this type of problem arises in the use of spherically symmetric models of the earth to obtain information on the structure of the mantle from known values of the torsional free oscillation frequencies; i.e. the inverse eigenvalue problem. One approach for solving this problem is to choose a set of possible structures and by finding how well the eigenfrequencies of each of these models match the desired values, the "best possible" model (or models) from the given set can be chosen. The difficulty is that the number of eigenvalues over the whole set of possible structures becomes quite large and so the time and computing resources spent in finding the model's eigenfrequencies needs to be optimised in terms of the information obtained.

What is needed to improve the error bounds is to give them a more global character, and one possible means for extending the bounds to show this is to refine the bound so that the dependence of C on κ becomes explicit. Thus for the purposes of comparison, we adopt

$$|\lambda_{\kappa} - \lambda_{\kappa}^{(N)}| \leq C N^{-p} \lambda_{\kappa}^q \quad ; N > N_0(\kappa)$$

(where C is now independent of both κ and N) as the standard error bound.

If we consider the bounds obtained in this form for a number of representative methods (Rayleigh-Ritz, Prüfer substitution, modified Prüfer substitution, finite difference and a similar operator approach) a number of features of the methods and the resulting approximations become apparent.

The first point is that since the rate of growth of the error of many methods is directly related to some high order derivative of the eigenfunction (the larger the order of convergence p , the higher the order of derivative) it can be preferable to use a method with a lower order of convergence to obtain better approximations of the larger eigenvalues at the expense of losing the high accuracy approximations

of the smaller eigenvalues.

The second point is that some methods (such as modified Prüfer substitution) perform better (i.e. the error behaves more uniformly) if the original eigenvalue problem is restated in Liouville normal form

$$(2.a) \quad -\ddot{v} + s v = \lambda v \quad t \in (0,1)$$

$$(2.b) \quad \begin{aligned} \alpha v(0) + \beta \dot{v}(0) &= 0 \\ \gamma v(1) + \delta \dot{v}(1) &= 0 \end{aligned}$$

$$(2.c) \quad v, \dot{v} \in C[0,1]$$

The final point is that for the considered approximation methods, it generally occurs that the improved uniformity of the approximation (if any is possible) is obtained at the expense of requiring stronger regularity conditions on the problem (e.g. the coefficients p and r should be sufficiently highly differentiable so that the problem can be transformed to normal form (2)). Thus it is desirable to find which methods will give the best results for a given set of regularity conditions. Although such an extensive analysis has not been undertaken, particular emphasis has been on the similar operator approach (based on transforming (1) to a quasi-normal form and then replacing the single coefficient function by a piece-wise constant approximation, the resulting problem is then analytically solvable) which gives uniform approximations under very weak regularity conditions.

THE NUMERICAL SOLUTION OF SECOND ORDER SYSTEMS OF ODES

Organizer: W. H. Enright
Dept. of Computer Science
University of Toronto

Panelists: D. G. Bettis
Dept. of Aerospace Engineering
and Engineering Mechanics
University of Texas at Austin

P. Deuflhard
Institut Fur Angewandte Mathematik
Universitat Heidelberg

F. T. Krogh
Jet Propulsion Laboratory
California Institute of Technology

The numerical solution of second order systems is one of the most important application areas for initial value methods. Special numerical methods for second order equations have been in use for over fifty years and these methods continue to be improved and developed in parallel with methods for first order systems. Two areas where these problems frequently arise and where special methods are heavily used are celestial mechanics and structural dynamics. Consequently the analysis and development of new methods is often tailored to classes of problems from these areas. Because of the gain in efficiency that is possible when the differential equation is independent of y' , there are generally two classes of methods developed for second order equations. The first class of methods are suitable for systems of the form:

$$y'' = f(x, y), \quad y(x_0) = y_0, \quad y'(x_0) = y'_0; \quad (I)$$

and the second class is suitable for general systems:

$$y'' = f(x, y, y'), \quad y(x_0) = y_0, \quad y'(x_0) = y'_0. \quad (II)$$

The members of this panel have each contributed to the improvement and development of numerical methods for second order problems. Professor Bettis has recently developed (with K. Horn) efficient embedded Runge-Kutta-Nystrom methods for second order systems (Bettis and Horn (1978)). He has also recently been involved in the development of exponentially fitted Runge-Kutta formula pairs. Professor Deuflhard has had considerable experience with extrapolation methods for second order equations and has recently developed a class of exponentially fitted extrapolation methods (Deuflhard (1978)). Dr. Krogh is well known for his work in the development of efficient multistep methods for systems of ODES. The multistep methods he has developed can handle

second order equations directly and he has investigated the advantages of doing so (Krogh (1975)). This discussion is to be an informal discussion of the state-of-the-art in this area. Each panelist will discuss aspects of this area that he has been involved in or finds to be of particular interest.

It is of course impossible to anticipate all the issues that will be addressed, but we will list a few of the issues and questions that are currently under active investigation and which are likely to arise. The first issue is the obvious one of detailing the advantages direct methods have over the alternative approach of reducing the system to a larger system of first order equations. Of course this is very much related to the general topic of comparing numerical methods (to be discussed in another panel session). The difficulties involved in comparing methods are particularly relevant here since we are not only interested in comparing different direct methods, but also in quantifying the advantage of direct methods.

A second topic which has recently received attention from several investigators (Gear (1978), Dahlquist (1978) and Thomas and Gladwell (1978)), is the question of stability of direct methods. Are there classes of problems where a large region of absolute stability is important, and if so, what is a suitable definition of stability for direct methods? This issue seems particularly crucial to problems arising in structural dynamics.

Another issue that is of considerable interest is the question of what an appropriate error control strategy should be for second order equations. Although this question is reasonably well understood for first order systems, the situation for second order systems is not as clear. Is it necessary to monitor and control the local error in y' as well as y for problems

of type I (or II). Some theoretical as well as computational investigations would be helpful here.

As a final example of issues that will likely be discussed, we will mention exponential fitting. This idea of choosing a parameter (or parameters) of the formula to match certain known characteristics of the solution is a topic that has received much attention in the literature for both first and second order equations. It is important to realize, that for this approach to be viable, one must take care in the selection and implementation of compatible error estimators and stepsize strategies. These components of a method must recognize the 'built-in' accuracy of the underlying formula and also account for the fact that the 'error coefficient' of the formula will now depend on the stepsize. It is also important to consider whether these formulas will generalize to systems of equations, and whether this will require an approximation to the matrix exponential and the derivation of formulas with matrix coefficients.

References:

- D.G. Bettis and M.K. Horn (1978) Embedded Runge-Kutta-Nystrom algorithms of order two through six, TCOM Report 78-3, Texas Institute for Computational Mechanics, University of Texas at Austin.
- G. Dahlquist (1978) On accuracy and unconditional stability of linear multistep methods for second order differential equations, BIT 18, pp. 133-136.
- P. Deuflhard (1979) A study of extrapolation methods based on multistep schemes without parasitic solutions, ZAMP, to appear.
- C.W. Gear (1978) The stability of numerical methods for second order ordinary differential equations, SINUM 15, pp. 188-197.
- F.T. Krogh (1975) Summary of test results with variants of a variable order Adams method, Computing Memorandum No. 376, Jet Propulsion Laboratory, Pasadena, California.
- R. Thomas and I. Gladwell (1978) Stability of some methods for second order ordinary differential equations with engineering applications, Numerical Analysis Report No. 29, Dept. of Mathematics, University of Manchester.

PANEL DISCUSSION ON NUMERICAL METHOD OF LINES SOFTWARE

G. D. Byrne
University of Pittsburgh

M. B. Carver
Atomic Energy of Canada Limited

A. C. Hindmarsh
Lawrence Livermore Laboratory

G. K. Leaf
Argonne National Laboratory

M. Minkoff
Argonne National Laboratory

W. E. Schiesser
Lehigh University

R. F. Sincovec
Boeing Computer Services, Inc.

1. Introduction (G. D. Byrne)

The numerical method of lines (NMOL) involves the following. A system of partial differential equations (PDE's) is discretized in its space-like variables to obtain a system of ordinary differential equations (ODE's) in the time-like variable, which is then solved by an ODE package. In a numerical method of lines package, the spatial discretization may be by a predetermined finite difference scheme, a user specified finite difference scheme of a certain general type, a Galerkin procedure in conjunction with B-splines, or a collocation procedure in conjunction with B-splines. In the case of Galerkin and collocation procedures, essential boundary conditions lead to algebraic equations or constraints, which are interspersed with the ODE's which must be solved. For the very simple heat equation, $u_t = u_{xx}$, finite difference-NMOL

involves the solution of a system of ODE's of the form $dy/dt = f(y,t)$, while collocation-B-spline or Galerkin-B-spline NMOL involves $A \cdot (dy/dt) = g(y,t)$ and the coefficient matrix A may be singular. Systems of these types of ODE's also arise from less simple PDE's for these discretizations. Of course, other discretizations are possible.

The panelists are associated with several general purpose NMOL software packages which use each of the discretization procedures cited above. Several of the panelists have also developed special purpose NMOL software for the solution of specific problems in heat conduction, oil reservoir simulation, and combustion. Some panelists have also been involved in the development of ODE software and in work with various nonlinear and linear systems solvers for use within the ODE software as in an NMOL package.

In the abstracts which follow, the participants are concerned with the requirements imposed on the ODE solver by the types of ODE's solved in

NMOL packages, desirable features of the ODE solver, the interface between the ODE solver and the rest of the NMOL code, the interface between the user and the NMOL code, and possible directions of research in packages for large, stiff systems of ODE's, which arise in the NMOL solution of multidimensional systems of PDE's.

2. The Ordinary Differential Equation Algorithm as Core Component of a Method of Lines Partial Differential Equation Software Package (M. B. Carver)

The properties required of an ODE integrator within a method of lines PDE package are different from those required of an integrator offered in a general subroutine library only in one respect: that the user is one more step removed from hands-on control of the integrator.

Even in a subroutine library, the average user lacks sufficiently specialized knowledge to guide integration by parameter selection. In either context, therefore, one requires sufficient robustness to automatically negotiate predictable difficulties without the guidance of a clairvoyant user, but enough scope for the informed user to provide tuning adjustments.

A number of efficient, well-documented, error-controlled, variable-step-size, integration algorithms are available, for example those discussed in references [1 to 4] and it is a prerequisite to provide such a routine as the core of any integration package. However, there are several specific areas where traditionally required user interaction can be either eliminated or reduced to optional status to improve effectiveness. They are:

- (a) Initial Step Size Selection: A routine with efficient step size adjustment is far better qualified than a user to provide this selection.

- (b) Tolerable Error Imposition: An algorithm normally accepts a step if the estimated local truncation error in Y_j satisfies $e_j < \text{TOL} * Y_{\text{base}_j}$, where Y_{base_j} is frequently defined as the maximum attained value of Y_j or made available as an array for user specification. Again users have no criteria on which to nominate each Y_{base_j} , and the automatic assignment $Y_{\text{base}_j} = \max(|Y_j|, Y_{\text{sig}})$ is general, and requires the user only to provide Y_{sig} , the value below which any $|Y_j|$ is effectively zero, if this differs from the machine unit round off.
- (c) Jacobian Evaluation and Handling: Most general integrators require estimates of the Jacobian matrix from time to time, and the user is often left to speculate on a choice of how these should be obtained. Sophisticates may gain efficiency by coding their own Jacobian, but within a method of lines package this possibility grows more remote, and could end up less efficient than numerical evaluation. Standard numerical procedure is to perturb each individual Y_j to obtain the set $\partial F_i / \partial Y_j$, requiring n function evaluation calls. This may be optimized; for example, a tridiagonal matrix Jacobian may be evaluated using only three calls, and proportionate savings may be realized in sparse matrices [5]. Sparse matrix Jacobian evaluation and handling optimized in this way is invaluable for general PDE/ODE packages in which structure is arbitrary. The mechanics are further discussed in a paper to this conference [6].
- (d) Discontinuity Detection and Negotiation: A switch in equation definition occurring at an unknown time dependent on critical relationships evolving during integration is a common physical phenomenon, but is traumatic for an integrator and frequently leads to time limit or termination. A robust integrator should detect such a discontinuity and transcend it issuing a suitable terse report. Optional user interaction to facilitate smoother handling of discontinuities should be made available, for example see reference [7].
- (e) Calling Sequence and Common Blocks: In a package, the user is shielded from direct communication through the integrator parameter list, but this is mentioned here to facilitate library or package interface design. Any integrator calling sequence requires ten parameters, the derivative subroutine, dependent variable and derivative vectors and their size, independent variable and interval, tolerance, warning flag, one working storage array and current step size. Only these parameters are mandatory and all other communication may be through optional common blocks. The integrator itself can assign parts of the working storage array in predictable sequence, and choose a suitable method for handling the Jacobian from an assessment of the equation structure and

available working storage.

The GEARZ integration algorithm used in the FORSIM[8] and MAKSIM[9] simulation packages combines the above consideration with the work described in references [2, 5 and 10].

References

- [1] C. W. Gear, The Automatic Integration of Ordinary Differential Equations, Comm. ACM, V14, 3, p. 176, March 1971.
- [2] A. C. Hindmarsh, The LLL Family of Ordinary Differential Equation Solvers, UCRL-78129, April 1976.
- [3] G. D. Byrne and A. C. Hindmarsh, A Poly-algorithm for the Numerical Solution of Ordinary Differential Equations, ACM TOMS, V1, p. 71, 1975.
- [4] L. F. Shampine, Practical Solution of Ordinary Differential Equations by Runge-Kutta Methods, SAND760585, 1976.
- [5] A. R. Curtis and J. K. Reid, FORTRAN Routines for the Solution of Sparse Sets of Linear Equations, AERE-R-6844, 1971.
- [6] M. B. Carver, In Search of a Robust Integration Algorithm, paper submitted to ACM SIGNUM Conference on Numerical Ordinary Differential Equations, 1979.
- [7] M. B. Carver and S. R. MacEwen, Analysis of a System Described by Implicitly Defined Ordinary Differential Equations Containing Discontinuities, Applied Mathematical Modelling, V2, 1978, p. 280.
- [8] M. B. Carver, D. G. Stewart, J. M. Blair and W. N. Selander, The FORSIM VI Package for Automated Solution of Arbitrarily Defined PDE/ODE Systems, Atomic Energy of Canada Limited report AECL-5821, February 1978.
- [9] M. B. Carver and A. W. Boyd, A Program Package Using Stiff Sparse Techniques for the Automatic Solution of Mass Action Chemical Kinetics, submitted to Int. J. Chem. Kinet.
- [10] M. B. Carver and A. P. Baudouin, Solution of Reactor Kinetics Problems Using Sparse Matrix Techniques in an ODE Integrator, Atomic Energy of Canada Limited report AECL-5177, 1975.
3. Method-of-Lines Considerations in ODE Software Design (A. C. Hindmarsh)

Among the most frequent uses of ordinary differential equation (ODE) software are those for the solution of partial differential equations (PDE's) by the numerical method of lines (NMOL). Typical of such problems are systems of parabolic PDE's in time t and a space vector x , given initial values at $t = 0$ and boundary conditions in

x. The resulting ODE problems are among the most challenging because of their potential size, complexity, and stiffness. PDE-based problems are frequently (but not always) stiff, either because of physical processes present (e.g. chemical kinetics), or because of the particular features of the spatial discretization (e.g. a simple 1-D diffusion equation finite-differenced on a fine mesh). The combination of size and stiffness imposes severe restrictions on the choice of ODE methods and on their implementation — restrictions that differ greatly from those for nonstiff problems.

Without going into the details of NMOL discretizations, the final result can be given in one of two forms. For finite difference approaches, one generally obtains an explicit ODE system $\dot{y} = f(y,t)$, where y is a vector of length N , and an initial value vector y_0 is known. For weighted residual methods (collocation, Galerkin, finite element, etc.), one gets instead a linearly implicit ODE system $A\dot{y} = g(y,t)$ where $A = A(y,t)$ is an $N \times N$ matrix (usually nonsingular) whose form depends on both the discretization method and the geometry of the problem. In the latter case, cost considerations require that the ODE algorithm treat the system in a direct manner, rather than invoke an algorithm for $\dot{y} = f$ with $f = A^{-1}g$.

When the ODE is stiff, accuracy and numerical stability dictate the use of implicit methods. Nearly all implicit methods, when implemented so as to retain the desired numerical stability, solve linear algebraic system that involve the $N \times N$ matrix $J = \partial f / \partial y$, or both A and $J = \partial g / \partial y$ in the case $A\dot{y} = g$. While the literature now contains numerous stiff ODE methods, most NMOL applications use methods based on the Backward Differentiation Formulas (BDF's). The reasons are partly historical, and partly based on the following facts about BDF methods:

1. They form a group of stiffly stable methods of various orders, easily allowing for variability of step size and order.
2. They require the solution of only one $N \times N$ nonlinear algebraic system on each time step, and so require only one copy of J or a related matrix (involving A , if present) at a time.
3. They do not need J (or A , if present) to be very accurate, and so do not require a reevaluation of J at every step.
4. They easily allow for sparse structure of J (and A , if present) to be utilized to gain both speed and storage economy.

Few other methods have all of these properties.

There are two ODE solvers which have seen particularly heavy use in the context of NMOL: The GEAR package uses (for stiff problems) fixed-step BDF's with interpolatory step changing. The newer EPISODE package resembles GEAR externally but is based on truly variable-step BDF's. Both take advantage of properties 1 to 3 above, using

a modified Newton (chord) iteration on the corrector, but handle only $\dot{y} = f$ and assume J to be dense. They save LU factorizations of the required matrix, and recompute this only as needed. In NMOL, J and A invariably have much sparse structure, and property 4 becomes crucial. To capitalize on it, variant versions of GEAR and EPISODE have been written, dictated by structures that commonly occur in PDE-based ODE systems. The modular design of the solvers easily allows the dense linear system treatment to be replaced accordingly. The implicit ODE case requires some alterations to other parts of the algorithm as well. The following variants exist currently:

1. GEARB treats J in band form. It is used in the NMOL packages PDEPACK, MOL1D, PDETWO, DSS/2, and FORSIM.
2. EPISODEB is the band variant of EPISODE, analogous to GEARB.
3. GEARBI treats J in blocked form and uses a block-iterative linear system algorithm (block-SOR). It is used in various 1-D and 2-D applications.
4. GEARS assumes a general sparse form for J and uses the Yale Sparse Matrix Package.
5. GEARIB treats the implicit ODE $A\dot{y} = g$ with A and $J = \partial g / \partial y$ in band form. It is used in the NMOL packages PDECOL and DISPL.
6. EPISODEIB is the variant of EPISODE analogous to GEARIB.

In the explicit ODE case, these solvers retain a nonstiff (Adams) option for convenience, and some of the NMOL packages relay this option back to the user.

Some applications have involved more special variants. For example, GEAR has been combined with a tailored preconditioned conjugate gradient algorithm for use on radiation diffusion-scattering problems.

Alternatives to chord iteration exist but have not been tried extensively in stiff ODE software. Experiments with nonlinear block-SOR iteration in an NMOL context have shown considerable promise. Another promising approach, in 2-D and 3-D, is to combine a BDF (or similar) method with a multiplicative matrix splitting of the type used in ADI.

4. Some Comments on the Impact of PDE Software on ODE Solvers (G. K. Leaf and M. Minkoff)

A common feature of software for solving parabolic differential equations is the use of the method of lines for the reduction of the partial differential equation to a system of ordinary differential equations. A typical PDE package will then use an ODE package to solve the resulting system of differential equations. The use of ODE solvers in conjunction with PDE software imposes certain requirements on the ODE

solver. First we note that the method of lines usually leads to systems of stiff ordinary differential equations; hence implicit time differencing is generally required. Such implicit method use Newton-type nonlinear equation solvers which require estimates for the Jacobian. Particularly in two spatial dimensions, the storage of this Jacobian becomes the dominant consideration in the implementation of PDE software. With the use of local approximations for spatial effects, the storage problem is ameliorated somewhat by the banded structure of the Jacobian. When a Galerkin type procedure is used, the resulting system of ODE's will be implicit in time. This makes it desirable to have solvers which can treat implicit time dependence in an efficient manner. When the method of lines is used, the boundary conditions associated with the PDE's lead to system of algebraic-differential systems. Finally, with PDE's a substantial cost savings can be achieved with the use of dump/restart features. Such features are almost mandatory in ODE solvers used in conjunction with PDE software.

5. Two Proposed Standard Interfaces to Facilitate the Use of NMOL Codes (W. E. Schiesser)

The remarkable flexibility and versatility of the numerical method of lines (NMOL) will most certainly facilitate the integration of systems of ordinary and partial differential equations over a spectrum of application areas. This is particularly true for scientists and engineers who do not wish to become experts in numerical analysis. If some general guidelines are developed for interfacing various components of a NMOL code, the user will be able to take better advantage of the flexibility of the method, and the exchange of applications between users will be facilitated. In this presentation, two interfaces will be considered:

- (1) The interface between the ODE integrator and the routines which approximate the spatial (boundary-value) derivatives in partial differential equations will be considered. In particular, the freedom to independently select the methods of integration for initial-value and boundary-value derivatives should be emphasized to users.
- (2) The interface between the general-purpose NMOL code and the routines which define the problem should be standardized as much as possible to facilitate the exchange of application problems. The essential elements (e.g., routines) to define a problem will be considered, and some suggestions given for a relatively standardized interface between the problem statement provided by the user and the NMOL code.

6. Comments on the Numerical Method of Lines (R. F. Sincovec)

The use of the numerical method of lines for the solution of one dimensional partial

differential equations is a remarkable success. Moreover, this approach to solving partial differential equations has allowed the development of some very useful and modestly reliable general purpose software for PDEs. In large measure these successes are due to the availability of good software for solving ordinary differential equations.

Unfortunately, one dimensional models are not adequate for simulating many physical phenomena and quite often one must use a model which is described or characterized by multidimensional partial differential equations. The fundamental concept of the numerical method of lines is essentially dimension independent, so in principle the technique may be directly applied to the higher dimensional problems. However, some of the practical computational aspects of the approach have prevented its widespread use of the higher dimensional problems which require the use of stiff ODE methods. For these problems, the limitations occur in the form of storage requirements and Jacobian matrix generation and solution time. The most frequently used stiff ODE techniques are those of Gear and the Jacobian matrix is required because a modified Newton method is used to solve the implicit nonlinear corrector equations. When a direct method is used to solve the resulting linear algebraic systems, the computation time per equation per time step increases drastically as the number of spatial dimensions increases. It would be extremely useful if reliable and efficient stiff ODE methods could be developed which do not require so much storage and computational cost per time step. In particular, it would be advantageous to use a nonlinear method, which requires only some portion of the associated Jacobian matrix. Alternatively, explicit methods with larger stability regions than those now in common use could be explored. It is conceivable that such ODE methods, which are basically different from the current backward differentiation formulas, could be developed - at least for some classes of problems.

Another approach would be to investigate and develop other solution techniques for the nonlinear corrector equations. The goal would be to find adequate nonlinear equation solution methods whose cost in both storage and execution time is less affected by dimensionality considerations. Nonlinear SOR-Newton, Newton-SOR or ADI techniques could be potential candidates. The use of any of these methods would probably impose restrictions on the type of problems which can be solved and this needs to be investigated. Also, the relationships between the convergence criteria and the error and stepsize control in the ODE solver would have to be explored.

Many problems encountered in practice can be reduced by the method of lines to a coupled differential-algebraic system. More attention needs to be paid to such systems. Finally, the ODE software itself should feature dynamic dimensioning.

PANEL ON TESTING

Moderator: T.E. Hull, Department of Computer Science, University of Toronto, Toronto, Canada

Panelists: M.B. Carver, Atomic Energy of Canada Limited, Chalk River, Ontario, Canada

W.H. Enright, Department of Computer Science, University of Toronto, Toronto, Canada

I. Gladwell, Department of Mathematics, University of Manchester, Manchester, England

F.T. Krogh, Jet Propulsion Laboratory, Pasadena, California, U.S.A.

L.F. Shampine, Sandia Laboratories, Albuquerque, New Mexico, U.S.A.

Each panelist was asked to write a brief summary of his point of view and/or any comments he would like to make on this topic, and the following is what was received:

M.B. Carver

There is no doubt that a program of testing will produce the most useful information if it investigates the widest possible spectrum of equation set type and size, accuracy level requested, range of magnitude of variables, etc., but, given a comprehensive test program and amassed results, the criteria used to assess or rate algorithms will depend heavily on the type of use for which an algorithm is to be selected. While a considerable market exists for extremely precise results, probably the greater use of integration algorithms is in general engineering and scientific calculations in which three to five reliable digits suffice.

Such algorithms usually reside in general sub-routine libraries or are embedded in user-oriented simulation packages, and the average user will not be equipped to select parameters to guide the integration algorithm. Thus the primary judgement criterion for such use is reliable performance in this accuracy range, and robust or self sufficient behaviour on encountering problems, such as discontinuities, which are common in practice but traumatic in integration.

W.H. Enright

During the past several years a group of us at Toronto have been involved in the development, assessment and comparison of numerical methods for ODE's. These investigations have convinced us of the need for a collection of testing programs to aid in the evaluation of numerical methods, and the design and use of such a package is the subject of this paper. The collection of programs actually consists of two very similar packages; one for the assessment of non-stiff methods and another for the assessment of stiff methods. Before we discuss the test programs in more detail we must describe the type of applications we envisage for it and distinguish two points of view that will be taken when interpreting the results.

With the number of new methods being developed and considered for publication, it is a formidable task for an author or a referee to evaluate a new method. In order to publish an algorithm in a

technical journal the author is expected to provide evidence of its performance characteristics compared to other methods. This evidence all too often takes the form of results of the new method relative to out-of-date methods on a few carefully selected problems. Refereeing such submissions is difficult and a referee is often reluctant to ask for more evidence since this often would require that an author obtain and implement a more appropriate reference method and run several new test problems. On the other hand, if a catalogue of standard benchmark results were available for existing software, and if these results could be duplicated and results for new methods readily obtained, then we could require much stronger evidence of performance for new methods. Another use of the testing package would be that of a potential user with a particular type of problem. He may wish to determine which method is most suitable for him by benchmarking the performance of available methods on a set of his own model problems.

There are two points of view that may be taken when assessing the performance of a numerical method. One may be primarily interested in the evaluation or assessment of the performance of an individual method, or alternatively he may be interested in comparing the relative performances of different methods. In the former case it is essential that the performance criteria and performance monitors that are introduced have no effect on the method being assessed. That is, we must ensure that the method is not modified in any way, as such modifications might degrade the performance of the method. In order to satisfy this condition the testing package must be flexible and able to assess a particular method on how well it (the method) is accomplishing what it was designed to accomplish.

On the other hand, when one is comparing methods, the various measures of reliability and efficiency can only be compared if the methods being compared are accomplishing the same task. One way to ensure that this is the case is to modify the methods so that they are all attempting to accomplish the same task. This was the approach adopted in Hull et al (1972), Enright et al (1975) and Enright et al (1976). This approach allows one to compare the relative performances of different techniques or approaches but it must be acknowledged that modified methods are compared, and these modifications might have a greater effect on some methods than others.

I. Gladwell

My interest in testing initial value solvers has arisen from a variety of causes. These include paying a semester visit to the University of Toronto, being asked questions by inexperienced users which I felt unable to answer without extensive testing, and also my involvement as contributor of the ordinary differential equations chapter of the NAG library. I have tested initial value solvers in two ways designed to be complementary to the many tests published in the literature. First with colleagues I have performed tests where the initial value solvers are treated as a black box and where I attempt to simulate their use by an inexperienced user for low accuracy computation. In these tests a variety of stiff and non-stiff solvers were tested and a selection of the test results are available in an internal report. Few conclusions are drawn in this report as too many anomalies arise in this type of testing. In a second set of tests, a research student and I have performed some simple profile tests (similar to those of Lyness and Kaganove for quadrature) on a small number of well-known subroutines. These tests do illustrate the effect of coding features of the routines, and it is hoped, in further tests, to attempt to modify the coding to improve the profile test results. The completed tests are summarized in a report which is in preparation.

F.T. Krogh

There are too many factors to be considered in the testing of ODE programs for me to attempt to summarize my views in a brief paragraph. The references: "On Testing a Subroutine for the Numerical Integration of Ordinary Differential Equations", J. ACM 20 (Oct. 1973), pp.545-562, and "Opinions on Matters Connected with the Evaluation of Programs and Methods for Integrating Ordinary Differential Equations", SIGNUM Newsletter 7 (Oct. 1972), pp.27-48, do a good job of representing my views. The following quote from the second of these is concerned with the issue I consider most important. "It is hoped that those who are carrying out evaluations for a wide audience will keep in mind the diversity of opinion likely to be found in that audience, and thus will present results in such a way that even those with a quite different point of view than the evaluator are able to draw conclusions from the results."

L.F. Shampine

I am most concerned with two aspects of testing. One is the question of comparability. Often codes do not have the same goals and even if they do, the way the task must be specified may be incompatible. Carelessness about the nature and purpose of tests lead to incorrect conclusions about the codes tested. The other aspect is, what are the tests supposed to prove? If a code performs well on a battery of problems, so what? We need theoretical insight to interpret and to design such tests. Analysis of models would also be useful for drawing better substantiated conclusions.

A USER INTERFACE STANDARD FOR ODE SOLVERS

Alan C. Hindmarsh
Lawrence Livermore Laboratory
Livermore, CA 94550

INTRODUCTION

During the last 10 to 15 years, a number of good general purpose Fortran programs have been developed for the solution of the initial value problem for systems of ordinary differential equations (ODE's). However, the various solvers differ substantially from one another, and the various versions of each individual solver often differ substantially from site to site. This lack of standardization causes inconvenience and confusion for users and maintenance headaches for the authors of the solvers.

At the same time, efforts undertaken in other areas of numerical analysis have succeeded in producing "systematized" collections (EISPACK, FUNPACK, LINPACK, MINPACK, etc.) of software of high quality and meeting certain uniformity standards. The idea of producing a similar systematized collection of ODE solvers (initial value problem), tentatively called ODEPACK, has been discussed since 1973 by various groups of people, from all over the world. Since 1976, efforts related to the ODEPACK concept have primarily involved only people at Dept. of Energy laboratories, and have been funded in part by the Applied Mathematical Sciences Research Program under the Office of Basic Energy Sciences in DOE.

A necessary preliminary to any actual development of an ODEPACK is the setting of standards for the interface between the user and the ODE solvers. Thus there has been considerable discussion of a detailed User Interface Standard (UIS) for ODEPACK. The user interface for a given solver consists largely of the call sequence of the subroutine which the user must call, together with the call sequences of the one or more user-supplied routines which the solver calls. The appropriate content of the user interface for an ODE solver varies greatly with the nature of the user and the nature of the solver. The goal of a UIS is to identify the features that are appropriate to all (or almost all) such contexts and put those features in a concrete form that is as simple as possible and yet meets the variety of demands envisioned. The UIS summarized below has evolved from numerous discussions and meetings, involving about 20 people.

The need to allow for a wide variety of solvers and user demands results in a UIS which includes more than call sequence arguments that

are to be standard across the ODEPACK collection and required of all users. It includes also items of communication that are solver-dependent and/or optional. Solver-dependent items are ones that will appear in the interface of some solvers, but not others. Optional items in a given solver interface are ones that the user may supply (if input) or access (if output) but is not required to. In the case of optional inputs, default values are always available. Solver-dependent items may or may not be optional, and optional items may or may not be solver-dependent. In all cases, these items are present in the UIS to assure that their names and meanings will be standard where and when they occur. The UIS is complicated further by the fact that communication need not be strictly through the call sequence: Optional inputs and outputs may be communicated by calls to other routines, by Common, or in some cases by allowing the user to supply his own version of one of the subroutines in the solver.

What follows is a summary of the present UIS. Complete details can be found in Ref. 1.

SUMMARY OF THE USER INTERFACE STANDARD

The primary means of communicating with the solver will be the call sequence of the subroutine associated with the solver. The parameters of this call sequence are summarized below.

F	= name of user-supplied subroutine defining the ODE system, $\dot{y} = f(t, y)$.
NEQ	= size of first-order ODE system.
Y	= array of dependent variables. Used to input initial values and output answers.
T	= independent variable. Used to input initial point and output the point at which Y is evaluated.
TOUT	= next value of t at which answers are desired.
ITOL, RTOL, ATOL	= input parameters for error control. Roughly, estimated local errors in y_i will be kept less than $RTOL(i) y_i + ATOL(i)$ in some norm. RTOL and ATOL can be of either scalar or vector type, and ITOL has one of 4 values to identify the type of each.

ITASK = input indicator for task desired:

- 1: normal calculation of $y(t)$ at $t = TOUT$ (possibly overshoot and interpolate).
- 2: take only one step and return.
- 3: hit TOUT exactly.
- 4: stop at the first internal mesh point t at or beyond TOUT.
- 5: take only one step, without passing TOUT.

ISTATE = state indicator (input and output),

Input:

- 1: first call (solver will initialize itself).
- 2: continue normally, with no change in inputs except TOUT and possibly ITASK.
- 3: continue normally, but with changes in other inputs.

Output:

- 1: nothing done ($TOUT = T$ on first call).
- 2: task was accomplished.
- 1: excess work on this call.
- 2: too much accuracy requested.
- 3: illegal input.
- ... (Other negative values are solver-dependent.)

A printed message will contain details of all returns with ISTATE < 0.

IOPT = input flag to signal use of optional inputs.

- 0: no optional inputs (defaults to be used).
- 1: all optional inputs will be examined.

RWORK, IWORK = real (single or double precision) and integer work space arrays. Length formulas will be given. The first few words of each may be used for optional communication.

LRW, LIW = user-declared lengths of RWORK, IWORK. (These will be checked.)

JAC = name of user-supplied subroutine to compute the Jacobian, $J = \partial f / \partial y$.

G = name of user-supplied function subprogram for the function g in the rootfinding problem $g(t, y(t)) = 0$.

Solver-dependent optional inputs and outputs may be communicated as parts of work arrays, by Common, or by optional calls to special routines. Optional inputs for which standard names and meanings exist include the maximum number of steps per call to the solver, the maximum number of f evaluations per call, the maximum order, the initial step size, and upper and lower bounds on step size. Optional outputs for which standard names and meanings exist are the current internal mesh point in t ; cumulative counts of steps and evaluations of f , J , and g ; the step size and order last used; the index of the largest weighted local error; and a tolerance scale factor to be used to continue after a request for too much accuracy was flagged.

There are two optional calls by the user to other routines that are defined in the standard. One is to reset the logical unit number for the printing of messages (if different from the default). The other is to suppress all printing of messages (the default being to print messages where appropriate).

Reference

- [1] A. C. Hindmarsh, A Tentative User Interface Standard for ODEPACK, LLL Report UCID-17954, October 1978.

In addition to the above parameters, which will be required of all solvers, there are solver-dependent parameters related to Jacobian matrices, rootfinding, method switches, etc. The following standard names will be used, where relevant:

AN INCOMPLETE LIST OF BETTER FORTRAN CODES FOR IVPS IN ODES

Robert D. Skeel
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

This listing is restricted to those computer programs which are available on magnetic tape or punched cards. First is a listing of codes grouped into collections and then a listing of software distributors. The author, who alone is responsible for the accuracy of this material, wishes to acknowledge the direct receipt of much useful information from the following people: M. Butler, G. D. Byrne, M. B. Carver, P. A. Fox, I. Gladwell, A. C. Hindmarsh, D. Kainer, C. B. Moler, D. Sayers, and L. F. Shampine.

PART I - COLLECTIONS OF CODES

1. International Mathematical and Statistical Libraries, Inc.--Chapter D: Differential Equations; Quadrature; Differentiation.
distributor: IMSL, Inc.

a. DGEAR

documentation: A. C. Hindmarsh, "GEAR: Ordinary differential equation solver," UCID-30001, Rev. 3, Lawrence Livermore Lab, U. of California/Livermore, December 1974.

description: modification of the Hindmarsh package GEAR based on Adams-Moulton and backward differentiation formulas of various orders.

b. DREBS

description: modification of the Bulirsch-Stoer Algol procedure DESUB based on rational extrapolation of Gragg's method.

c. DVERK

documentation: T. E. Hull, W. H. Enright, and K. R. Jackson, "User's guide for DVERK--A subroutine for solving non-stiff ODE's," T.R. No. 100, D. of C.S., U. of Toronto, Oct. 1976.

description: based on 8-stage Runge-Kutta formula of orders 5 and 6 developed by J. H. Verner.

2. NAG FORTRAN Library, Mark 7--Chapter D02: Ordinary Differential Equations.
distributor: Numerical Algorithms Group.

a. D02AFF

description: Least-squares fitting of a series of Chebyshev polynomials to solve a system of linearized ODEs.

2. b. D02BAF/D02BBF/D02BDF/D02BGF/D02BHF/D02PAF, D02XAF/D02PAF, D02XBF
description: based on Merson's 5-stage Runge-Kutta formula of order 4; various drivers perform output at user selected points, global error estimation, stiffness checks, root-finding.

c. D02CAF/D02CBF/D02CGF/D02CHF/D02QAF, D02XGF/D02QAF, D02XHF

description: essentially GEAR, Rev. 3 of Hindmarsh with Adams-Moulton formulas selected; drivers for various tasks.

d. D02EAF/D02EBF/D02EGF/D02EHF/D02QBF, D02XGF/D02QBF, D02XHF

description: essentially GEAR, Rev. 3 of Hindmarsh with backward differentiation formulas selected; drivers for various tasks.

e. D02YAF

description: step-oriented integrator based on Merson's 5-stage Runge-Kutta formula of order 4.

††f. D02AAF (superseded by D02YAF)

††g. D02ABF (superseded by D02BAF and D02PAF)

††h. D02AHF (superseded by D02CAF and D02QAF)

††i. D02AFJ (superseded by D02EAF and D02QBF)

3. Bell Laboratories Mathematical Subroutine Library, PORT--differential equations chapter
distributor: Bell Laboratories.

a. ODES/ODES

documentation: N. L. Schryer, "A user's guide to ODES, a double precision ordinary differential equation solver," Computing Sci. T.R. #33, Bell Laboratories, Murray Hill, N.J., 1975.

description: based on extrapolation of Gragg's method.

note: uses the function subprogram ILMACH, the storage allocator and the error handling facility of the PORT library.

†† This routine will be withdrawn at Mark 8.

4. Collected Algorithms from ACM--category D2:
Ordinary Differential Equations
distributor: ACM Algorithms Distribution
Service, c/o IMSL, Inc.
 - a. GERK
other distributor: NESC (part of MATHLIB,
No. 654)
documentation: L. F. Shampine and H. A.
Watts, "Algorithm 504 GERK: Global Error
Estimation for Ordinary Differential
Equations [D]," *ACM Trans. on Math.
Software* 2, 2 (June 1978), 200-203.
description: the 6-stage Runge-Kutta-Fehlberg
formula of order 4 and 5 with global error
estimation by Richardson extrapolation.
 - b. STINT
documentation: J. M. Tendler, T. A. Bickart,
and Z. Picel, "Algorithm 534 STINT:
Stiff (differential equations) INTEgrator
[D2]," *ACM Trans. on Math. Software* 4, 4
(Dec. 1978), 399-403.
Z. Picel, T. A. Bickart, and J. M.
Tendler, "STINT, Stiff ordinary differ-
ential equations INTEgrator, Program and
user manual," TR-76-12, D. of Elect. &
Comp. Eng., Syracuse U., Syracuse, N.Y.,
Dec. 1976.
description: cyclic composite multistep
methods of orders 1 through 7, particu-
larly well suited for systems wherein
the Jacobian matrix has complex eigen-
values near the imaginary axis.
5. DEPAC
description: systematized collection of ODE
solvers being developed at Sandia Laboratories
 - a. RKF45
distributors: Prof. Cleve Moler (for the
complete set of subroutines in F, M&M)
NESC (part of MATHLIB, No. 654)
documentation: G. E. Forsythe, M. A. Malcolm,
and C. B. Moler, *Computer Methods for
Mathematical Computations*, Prentice-Hall,
Englewood Cliffs, N.J., 1977.
L. F. Shampine and H. A. Watts,
"Practical solution of ordinary differ-
ential equations by Runge-Kutta
methods," SAND 76-0585, Sandia Labs.,
Albuquerque, N.M., 1976.
description: based on the 6-stage Runge-
Kutta-Fehlberg formula of order 4 and 5.
 - b. GERK
see 4a.
 - c. ODE/DE/STEP,INTRP
distributor: NESC (No. 640; also part of
MATHLIB, No. 654)
documentation: L. F. Shampine and M. K.
Gordon, *Computer Solution of Ordinary
Differential Equations*, Freeman, San
Francisco, Calif., 1975.
description: variable-order variable-step
Adams PECE code with local extrapolation.
 - d. ODERT/DE/STEP,INTRP
distributor: NESC (No. 640)
description: root-finder.
5. e. RKSU
distributor: Dr. L. F. Shampine
documentation: L. F. Shampine and J. A.
Wisniewski, "The variable order Runge-
Kutta code RKSU and its performance,"
SAND78-1347, Sandia Labs., Albuquerque,
N.M., July 1978.
description: based on a 5-stage RK formula
of orders 3 and 4 and a 13-stage formula
developed by J. H. Verner of orders 7
and 8.
6. LLL family of ordinary differential equation
solvers
 - a. GEAR
distributor: NESC (No. 592)
documentation: A. C. Hindmarsh, "GEAR:
Ordinary differential equation system
solver," UCID-30001, Rev. 3, Lawrence
Livermore Lab., U. of Calif./Livermore,
Dec. 1974.
description: revision of the Gear subroutine
DIFSUB based on Adams-Moulton and back-
ward differentiation formulas of various
orders.
 - b. GEARB
distributor: NESC (No. 661)
documentation: A. C. Hindmarsh, "GEARB:
Solution of ordinary differential
equations having banded Jacobian,"
UCID-30059, Rev. 1 with corrections,
L.L.L., July 1976.
 - c. GEARS
distributor: Dr. A. C. Hindmarsh
documentation: the source is self-document-
ing as to usage.
description: uses Yale Sparse Matrix
Package; intended for ODEs having a
sparse Jacobian matrix.
 - d. GEARIB
distributor: Dr. A. C. Hindmarsh
documentation: A. C. Hindmarsh, "Prelimin-
ary documentation of GEARIB: Solution
of implicit systems of ordinary differ-
ential equations with banded Jacobian,"
UCID-30130, L.L.L., Feb. 1976.
 - e. GEARBI
distributor: Dr. A. C. Hindmarsh
documentation: A. C. Hindmarsh, "Prelimin-
ary documentation of GEARBI: Solution
of ODE systems with block-iterative
treatment of the Jacobian," UCID-30149,
L.L.L., Dec. 1976.
description: GEAR variant using block-SOR
solution of linear systems.

6. f. EPISODE

distributor: NESC (No. 675)

documentation: A. C. Hindmarsh and G. D. Byrne, "Applications of EPISODE: an Experimental Package for the Integration of Systems of Ordinary Differential Equations," in *Numerical Methods for Differential Systems*, L. Lapidus and W. E. Schiesser, eds., Academic Press, 1976, 147-166.

A. C. Hindmarsh and G. D. Byrne, "EPISODE: an Effective Package for the Integration of Systems of Ordinary Differential Equations," UCID-30112, Rev. 1, L.L.L., April 1977.

description: variable-order code based on *variable-step* Adams-Moulton and backward differentiation formulas.

g. EPISODEB

distributor: NESC (No. 705)

documentation: G. D. Byrne and A. C. Hindmarsh, "EPISODEB: an Experimental Package for the Integration of Systems of Ordinary Differential Equations with Banded Jacobians," UCID-30132, L.L.L., April 1976.

h. EPISODEIB

distributor: Dr. A. C. Hindmarsh

documentation: G. D. Byrne, H. G. Brown, and A. C. Hindmarsh, "EPISODEIB: an Effective Package for the Integration of Systems of Ordinary Differential Equations of Implicit Banded form," in *Extended Abstracts, Texas Conf. on Math. Software*, Univ. of Texas at Austin, March 1978.

7. STFODE/COLODE

distributor: NESC (No. 652).

developers: B. L. Hulme and S. L. Daniel.
documentation: SAND74-0380, SAND74-0381,
Dec. 1974.

description: collocation methods for solving stiff ODEs.

8. STIFFZ/GEARZ

distributor: Dr. M. B. Carver

developers: M. B. Carver and D. G. Stewart
documentation: AECL FTN LIBRARY description
sheets for STIFFZ numbered 1-17-20 and
dated Sept. 1978.

description: modifications of the GEAR routines of A. C. Hindmarsh, L.L.L., and the sparse matrix routines of A. C. Curtis and J. K. Reid, Harwell.

9. RK58

distributor: Prof. F. E. Cellier

documentation: complete documentation as
header in the subroutine.

description: based on the 6-stage Runge-Kutta-Fehlberg formula of orders 4 and 5, as well as on the 13-stage RKF formula of orders 7 and 8; user is allowed to specify sequences of termination criteria as crossings through zero.

1. IMSL, Inc.

Sixth Floor, GNB Building
7500 Belaire Boulevard
Houston, Texas 77036

2. The Secretary, NAG USA Inc.

1250 Grace Court
Downers Grove, Illinois 60515
312/971-2337

outside North America:

The NAG Library Service Coordinator
Numerical Algorithms Group Ltd.
7 Bandbury Road
Oxford OX2 6NN United Kingdom
Oxford (0865) 511245

3. Bell Laboratories

Computing Information Service
600 Mountain Avenue
Murray Hill, New Jersey 07974

The above address is for a nonprofit educational institute interested in obtaining a royalty-free license for the PORT Library. Inquiries for a commercial license should be sent to

Western Electric Co.

Patent Licensing Manager

P. O. Box 20046

Greensboro, North Carolina 27420

4. National Energy Software Center (NESC)

Argonne National Laboratory
9700 South Cass Avenue
Argonne, Illinois 60439

how to obtain code: contact your institutional
NESC representative or computer center

5. Professor Cleve Moler

Department of Mathematics
University of New Mexico
Albuquerque, New Mexico 87131

6. Dr. L. F. Shampine

Numerical Mathematics Division 5642
Sandia Laboratories
Albuquerque, New Mexico 87185

7. Dr. A. C. Hindmarsh

Mathematics and Statistics Section, L-300
Lawrence Livermore Laboratory
P. O. Box 808
Livermore, California 94550

8. Dr. M. B. Carver

Mathematics and Computation Branch
Chalk River Nuclear Laboratories
Chalk River, Ontario
Canada K0J 1J0

9. Professor F. E. Cellier

Institute for Automatic Control
The Swiss Federal Institute of Technology
ETH-Zentrum
CH-8092 Zurich Switzerland

U. S. ATOMIC ENERGY COMMISSION
UNIVERSITY-TYPE CONTRACTOR'S RECOMMENDATION FOR
DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

1. AEC REPORT NO. COO-2383-0058	2. TITLE WORKING PAPERS FOR THE 1979 SIGNUM MEETING ON NUMERICAL ORDINARY DIFFERENTIAL EQUATIONS
--	--

3. TYPE OF DOCUMENT (Check one):

☒ a. Scientific and technical report

☐ b. Conference paper not to be published in a journal:

Title of conference _____

Date of conference _____

Exact location of conference _____

Sponsoring organization _____

☐ c. Other (Specify) _____

4. RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

☒ a. AEC's normal announcement and distribution procedures may be followed.

☐ b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.

☐ c. Make no announcement or distribution.

5. REASON FOR RECOMMENDED RESTRICTIONS:

6. SUBMITTED BY: NAME AND POSITION (Please print or type)

C. W. Gear
Professor and Principal Investigator

Organization

Department of Computer Science
University of Illinois U-C
Urbana, IL 61801

Signature 	Date March 1979
--	--------------------

FOR AEC USE ONLY

7. AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION

8. PATENT CLEARANCE:

☐ a. AEC patent clearance has been granted by responsible AEC patent group.

☐ b. Report has been sent to responsible AEC patent group for clearance.

☐ c. Patent clearance not required.

BIBLIOGRAPHIC DATA SHEET		1. Report No. UIUCDCS-R-79-963	2.	3. Recipient's Accession No.
4. Title and Subtitle WORKING PAPERS FOR THE 1979 SIGNUM MEETING ON NUMERICAL ORDINARY DIFFERENTIAL EQUATIONS			5. Report Date March 1979	6.
7. Author(s) R. D. Skeel, Editor			8. Performing Organization Rept. No. 79-963	
9. Performing Organization Name and Address Department of Computer Science University of Illinois U-C Urbana, IL 61801			10. Project/Task/Work Unit No.	
			11. Contract/Grant No.	
12. Sponsoring Organization Name and Address National Science Foundation Washington, DC			13. Type of Report & Period Covered working papers	
			14.	
15. Supplementary Notes				
16. Abstracts This report gives a summary of the papers presented at the meeting. It consists of all working papers distributed at the conference and all working papers received too late for distribution. In addition, abstracts and/or summaries are included where practical for those talks and workshop sessions that did not generate papers. It is hoped that this document will be useful reference to very current research in ODEs. These papers are working papers, that is they are preliminary versions of papers that will be submitted for publication. Out of courtesy to their authors, the papers should not be indiscriminantly reproduced.				
17. Key Words and Document Analysis. 17a. Descriptors numerical ordinary differential equations 1979 SIGNUM Meeting				
17b. Identifiers Open-Ended Terms				
17c. COSATI Field/Group				
18. Availability Statement unlimited			19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages
			20. Security Class (This Page) UNCLASSIFIED	22. Price



UNIVERSITY OF ILLINOIS-URBANA



3 0112 000487675